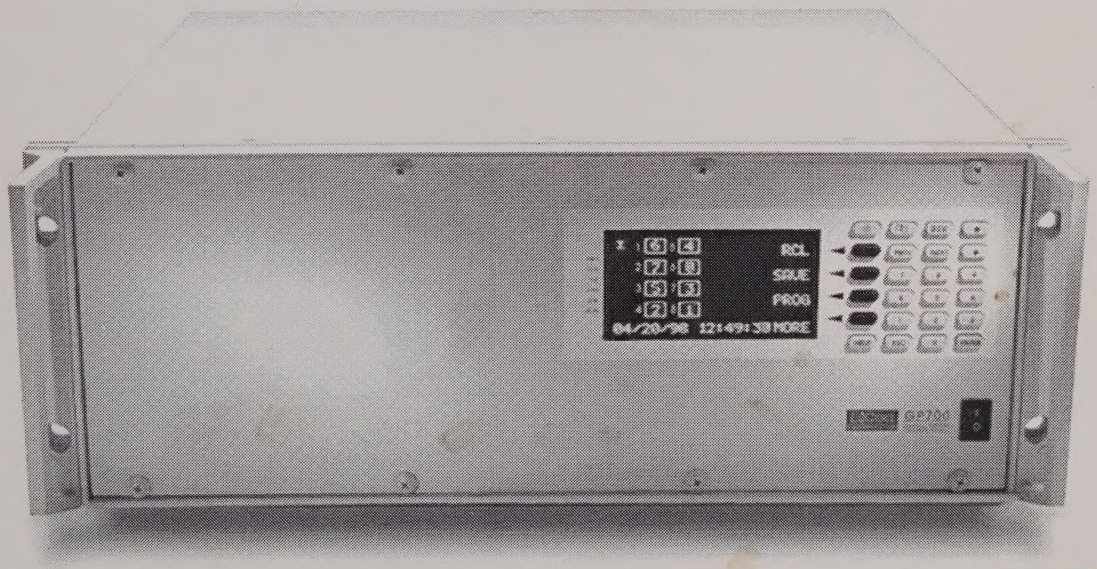
DiCon
FIBEROPTICS, INC

GP700 Programmable Fiberoptic Instrument

Operation Manual



MP MASTER
MA26207

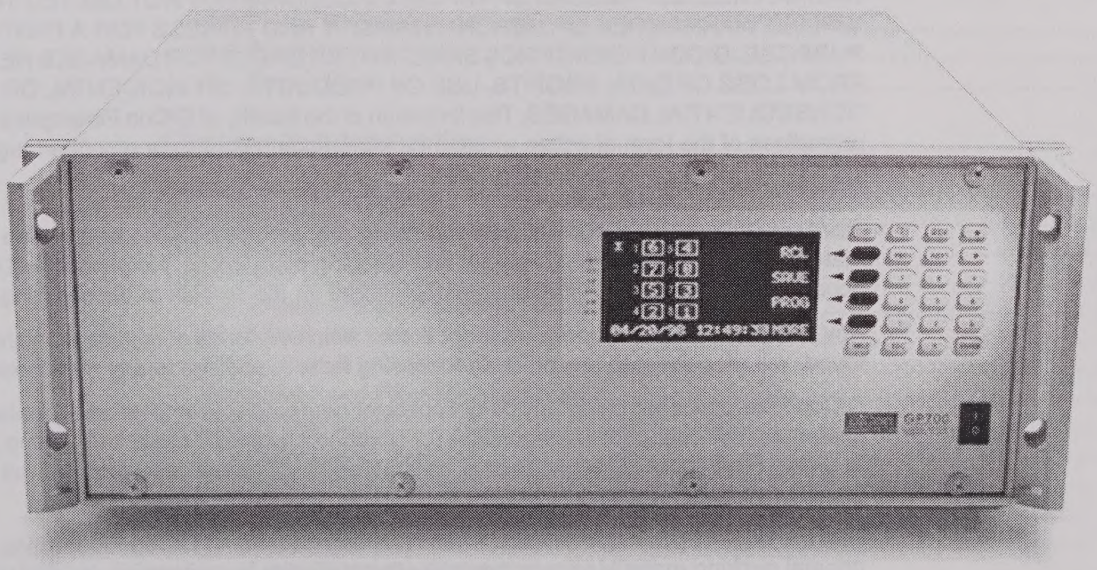
Digitized by the Internet Archive
in 2026

https://archive.org/details/manualsplus_12724

DiCon
FIBEROPTICS, INC

GP700 Programmable Fiberoptic Instrument

Operation Manual



91506 Revision B

Copyright © 1998 DiCon Fiberoptics, Incorporated.
All rights reserved. Printed in the United States of America.

No part of this manual may be reproduced, in any form or by any means, electronic or mechanical, including photocopying, without the express written permission of DiCon Fiberoptics.

We have reason to believe that a number of the company and product names appearing herein constitute trademarks or have been designated as such by their respective holders. No attempt has been made to designate these words as trademarks or as registered trademarks. The inclusion, or exclusion of a word or term is not intended to affect, or express a judgment on, the validity or legal status of the word or term as a trademark, or other proprietary term.

The information provided within this manual has been carefully reviewed for technical accuracy. DiCon Fiberoptics reserves the right to correct technical or typographical errors at any time, without prior notice. In no event shall DiCon Fiberoptics be liable for errors contained herein, or for incidental or consequential damages arising out of or related to, this document or the information contained in it.

Warranty

DiCon Fiberoptics warrants this product to be free from defects in both workmanship and performance for a period of one year from the time of original shipment. During the warranty period, DiCon Fiberoptics will, at its option, repair or replace any material found to be defective.

The foregoing warranty extends to all cases, except where the product has been damaged through misuse, mishandling, inadequate maintenance, owner modification, failure to follow the installation and operating instructions provided by DiCon Fiberoptics, flood, fire or other events outside our reasonable control.

EXCEPT AS SPECIFIED HEREIN, DICON FIBEROPTICS MAKES NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. DICON FIBEROPTICS SHALL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES. This limitation of the liability of DiCon Fiberoptics will apply regardless of the form of action whether based on contract, tort, or any other legal theory.

All Returns

Contact DiCon Fiberoptics before returning any product. DiCon will provide a Return Material Authorization (RMA) number and shipping instructions. No product will be accepted without an RMA number clearly marked on the outside of the shipping material.

Any unit that is returned under warranty, but for which evidence of misuse or mishandling is found, will be subject to testing and processing fees, in addition to any repair costs.

DiCon Fiberoptics will cover any freight costs for returning warranty returned material to the customer. The customer is responsible for covering any freight costs for sending materials to DiCon Fiberoptics. The customer is responsible for all freight costs for returned materials found to be out of warranty.

Use the original packing materials when returning material to DiCon Fiberoptics. If the original packing materials are unavailable, the customer is responsible for ensuring adequate packing materials are used to prevent damage occurring during shipping to DiCon Fiberoptics.

Contents

List of Figures ..... *ix*

List of Tables ..... *xi*

About This Manual ..... *xiii*

Quick Start ..... *xiii*
Manual Organization ..... *xiii*
Conventions ..... *xiv*

Chapter 1: Unpacking and Setting Up 1

Unpacking 1
Handling Fiberoptic Cables and Connectors 2
- Storing Optical Connectors 2
- Cleaning Optical Connectors 2
- Mating Optical Connectors 3

Rackmount Guidelines 3
Applying Power 3
- AC Power Requirements 3
- AC Fuse Requirements 4
- AC Power Cord 4

Chapter 2: Introduction 5

Product Overview 5
The Front Panel 6
The Keypad 6
The Display 6
- Two-Position Switch Module Status 7
- Multi-Channel Switch Module Status 8
- Matrix Switch Module Status 8
- Attenuator and Filter Module Status 9
- Status Indicators 9
  - Busy Status Indicator 9
  - Error Status Indicator 10
  - Remote Interface Status Indicator 10
  - Talk Interface Status Indicator 10
  - Listen Interface Status Indicator 10
  - Service Request Interface Status Indicator 10
- Softkey Labels 10
- The Prompt Line 10

The Rear Panel 11

GPIB Receptacle 11
RS-232 Receptacle 11
External Trigger Receptacle 11
General Purpose Output Receptacle 11
Optical Adapters and Feedthroughs 12

Chapter 3: Front-Panel Operation 15

Starting the GP700 15
Operating the GP700 15
Getting Help 16
Returning to the Main Menu 17
Changing Display Brightness 17
Changing Module Settings 17
Saving a Device State 18
Saving a Complete Device State 18
Saving a Partial Device State 18
Recalling a Device State 18
Recording a Mini-Program 19
Running a Mini-Program 19
Configuring the Input/Output Ports 19
Changing the GPIB Address 20
Changing the RS-232 Baud Rate 20
Setting the Clock 20

Chapter 4: Mini-Programs 21

Overview 21
Mini-Program Example 21
Recording a Mini-Program 24
Changing a Module Setting 26
Recalling a Device State 27
Waiting for a Trigger 27
Setting Up a Program Loop 27
Sending a General Purpose Output 28
Saving a Mini-Program 28
Exiting Record Mode Without Saving 28
Running a Mini-Program 28
Mini-Program Stops – Wait Encountered 28
Mini-Program Errors 29
Aborting a Mini-Program 29

Chapter 5: Remote Operation 31

Remote Interfaces 31
GPIB Interface Programming 32
GPIB Basics 32
Talkers, Listeners, Controllers, and Devices 32
The System Interface 32
Managing Information on the Bus 32
GPIB Command Structure 32
GPIB Addressing 33
GPIB Terminator 33
GPIB Queries 34
IEEE 488 Compliance Criteria 34

RS-232 Interface Programming 35
RS-232 Basics 35
What's On The Interface? 35
The System Interface 35
Managing Information on the Interface 35
RS-232 Interface Configuration 35
RS-232 Rear Panel Receptacle Configuration 36
RS-232 Command Structure 37
RS-232 Terminator 37
RS-232 Queries 37
Error Messages 38
Capturing Error Events 38
Local, Remote, and Local Lockout Modes 39
Command Timing 39
Calibrating Setting Time 40
Implementing Mini-Programs Remotely 42

Chapter 6: Remote Commands 43

Command Set Summary 43
IEEE488 Common Commands 43
GP700 Device-Specific Commands 44
Remote Mini-Program Instructions 45
Configuration Assumptions 46
GPIB Control 46
RS-232 Control 46
Command Conventions 47
Names 47
Availability 47
Parameters 47
Command Definitions 48
Attenuation Level Select 48
Attenuation Level Query 48
Begin Loop 49
Clear Status 49
Decrement M-Type Module Channel 50
Turn Off Display 50
Turn On Display 51
End Loop 51
Event Status Enable 52
Event Status Enable Query 53
Event Status Register Query 53
External Trigger Configuration 54
External Trigger Configuration Query 55
Filter Center Wavelength Select 55
Filter Center Wavelength Query 56
General Purpose Output Configuration 56
General Purpose Output Configuration Query 57
General Purpose Output 58
Matrix Channel Select 58
Matrix Channel Query 58
Identification Query 59
Increment M-Type Module 59
Select M-Type Module Channel 60

M-Type Module All Call 60
M-type Module Channel Query 61
Mini-Program Begin 62
Mini-Program End 62
Operation Complete 63
Operation Complete Query 63
Abort Mini-Program 64
Delete Mini-Program 64
Execute Mini-Program 65
Append New Line to Mini-Program 65
Program Contents Query 66
Recall Device State 66
Reset 67
S-Type Module Channel Select 68
S-Type Module All Call 68
S-Type Module Channel Query 70
Save Device State 70
Close RS-232 Interface 71
Open RS-232 Interface 71
Service Request Enable 71
Service Request Enable Query 72
Status Byte Query 73
System Configuration Query 74
Set System Date 75
System Date Query 75
System Error Query 76
Set System Time 77
System Time Query 77
Toggle S-Type Module 78
Trigger 78
Self-Test Query 79
Wait-To-Continue 79
Wait For Trigger Input 80

Chapter 7: Servicing and Troubleshooting 81

Performing a Functional Test 81
Investigating Common Problems 81
Interpreting Remote Error Codes 83
Improving Repeatability 84
Warm Up Source, Detector, and Switch 84
Use An Ultra-Stable Light Source 84
Avoid Source-Destabilizing Techniques 85
Minimize Back-Reflection 85
Use an Ultra-Stable Detector 85
Use Low-PDL Components 86
Use a Detector With Low Polarization Sensitivity 86
Monitor Source Drift 86
Maintain Stable Temperature 86
Use Single-Channel Sequential Switching 86
Minimize Connectors and Couplers 87
Eliminate Fiber Stress 87
Eliminate Measurement Aberrations 88
Contacting DiCon Fiberoptics 88

Appendix A: Specifications A-1

Regulatory Conformance A-1
Year 2000 Statement A-1
About Specifications A-1
Mechanical Specifications A-2
Housing Diagrams A-3
Electrical Specifications A-7
Environmental Specifications A-8
Optical Specifications A-8
A-Type Attenuator Modules A-8
F-Type Filter Modules A-9
I-Type Matrix Switch Modules A-10
M-Type Multi-Channel Switch Modules A-11
S-Type Two-Position Switch Modules A-12

Appendix B: Modules B-1

Two-Position Switch Modules B-1
Two-Position Switch Module Synchronization B-2
Multi-Channel Switch Modules B-3
Simplex 1×N Configurations B-3
Synchronous Duplex and Triplex 1×N Configurations B-4
2×N and 3×N Non-Blocking Configurations B-4
2×N and 3×N Blocking Configurations B-5
Multi-Channel Switch Module Synchronization B-5
Matrix Switch Modules B-6
Attenuator Modules B-6
Filter Modules B-8

Appendix C: Sample Configurations C-1

Appendix D: Device Compatibility D-1

Figures

Chapter 1: Unpacking and Setting Up

1-1. AC Power Inlet Fuse Drawer 4

Chapter 2: Introduction

2-1. GP700 Front Panel (4U Rackmount Chassis) 6
2-2. GP700 Front-Panel Display 7
2-3. Front-Panel Display With Multiple M- and S-Type Modules. 7
2-4. Front-Panel Display With 3×5 M-Type Module 8
2-5. Front-Panel Display With I-Type Module 9
2-6. Front-Panel Display With A- and F-Type Modules 9
2-7. Rear Panel of the GP700 (Rackmount Chassis) 11
2-8. Pigtail Feedthroughs 12
2-9. Connectors and Adapters 13

Chapter 3: Front-Panel Operation

3-1. GP700 I/O Ports 19
3-2. I/O Port Trigger Signals 20

Chapter 4: Mini-Programs

4-1. Example Test Set-Up 22
4-2. Record-Mode Front-Panel Display 25

Chapter 5: Remote Operation

5-1. RS-232 Receptacle 36
5-2. RS-232 Cable Configuration 36
5-3. Set Command Timing 41

Chapter 6: Remote Commands

6-1. Sample Application Summary Box 47

Appendix A: Specifications

A-1. 2U Standard Rackmount Chassis A-3
A-2. 4U Standard Rackmount Chassis A-4
A-3. 6U Extended Rackmount Chassis A-5
A-4. Benchtop Chassis A-6

Appendix B: Modules

B-1. On/Off Switch Module B-1
B-2. Duplex On/Off Switch Module B-1
B-3. 1×2 Switch Module B-1
B-4. Duplex 1×2 Switch Module B-2
B-5. 2×2 Switch Module B-2
B-6. Multi-Channel Switch Module Mechanical Schematic B-3
B-7. Simplex 1×6 Switch Module B-3
B-8. Synchronous Triplex 1×5 Switch Module B-4
B-9. Non-Blocking 3×10 Switch Module B-5
B-10. Blocking 3×12 Switch Module B-5
B-11. 6×8 Matrix Switch Module B-6
B-12. Attenuator Module Mechanical Schematic B-7
B-13. Attenuator Module B-7
B-14. Filter Module Mechanical Schematic B-8
B-15. Filter Module B-8

Appendix C: Sample Configurations

C-1. Multiple Filter Modules C-1
C-2. Multiple Two-Position Switch Modules and an Attenuator Module C-2
C-3. A Multi-Channel Switch Module and an Attenuator Module C-3
C-4. Multiple Switch Modules and Couplers C-4
C-5. Switch Modules with Internal Connections C-5
C-6. Switch and Attenuator Modules with Internal Connections C-6

Tables

Chapter 1: Unpacking and Setting Up

1-1. A/C Power Requirements 3

Chapter 2: Introduction

2-1. GP700 Module Types 5

Chapter 3: Front-Panel Operation

3-1. Main-Menu Permanent Keys 16
3-2. Main-Menu Softkeys 16

Chapter 4: Mini-Programs

4-1. Sample Program Initialization Sequence 22
4-2. Sample Program Entry Sequence 23
4-3. Keystroke Comparison of Non-Automated Versus Automated Testing 24
4-4. Mini-Program Record-Mode Key Functions 25
4-5. Mini-Program Wait Options 27

Chapter 5: Remote Operation

5-1. GP700 IEEE488 Specification Compliance 34
5-2. RS-232 Configuration Parameters 35
5-3. RS-232 Line Assignments 36
5-4. Command Synchronization Using General Purpose Outputs 42

Chapter 6: Remote Commands

6-1. IEEE488 Common Command Set 43
6-2. GP700 Device-Specific Command Set 44
6-3. Remote Mini-Program Instruction Set 45
6-4. Standard Event Status Enable Register 52
6-5. Standard Event Status Register 54
6-6. Soft Reset State 67
6-7. Two-Position Switch Module Actuation 68
6-8. Typical Two-Position Switch Module Banking Configurations 69
6-9. Service Request Enable Register 72
6-10. Status Byte Register 73
6-11. Mini-Program Wait Options 80

Chapter 7: Servicing and Troubleshooting

7-1. Troubleshooting Guide 81
7-2. Remote Error Codes 83

Appendix A: Specifications

A-1. Mechanical Specifications A-2
A-2. Module Weight A-2
A-3. Electrical Specifications A-7
A-4. Environmental Specification A-8
A-5. Attenuator Range-Dependent Specifications A-8
A-6. Attenuator General Performance Specifications A-8
A-7. Filter Optical Performance A-9
A-8. Matrix Switch Singlemode Optical Performance A-10
A-9. Matrix Switch Multimode Optical Performance A-10
A-10. Multi-Channel Switch Singlemode Optical Performance A-11
A-11. Multi-Channel Switch Multimode Optical Performance A-11
A-12. Two-Position Switch Singlemode Optical Performance A-12
A-13. Two-Position Switch Multimode Optical Performance A-12

Appendix B: Modules

B-1. Typical Two-Position Switch Banking Configurations. B-2

Appendix D: Device Compatibility

D-1. Porting GPIB Code Written for Previous DiCon Switch Models D-1
D-2. Porting RS-232 Code Written for Previous DiCon Switch Models D-2

About This Manual

Quick Start

For a quick overview of the GP700, read Chapter 1, "Unpacking and Setting Up", Chapter 2, "Introduction", and Chapter 3, "Front-Panel Operation".

Manual Organization

Chapter 1, "Unpacking and Setting Up", presents guidelines for handling fiberoptic cables and connectors and instructions for connecting your GP700 to a power source.

Chapter 2, "Introduction", summarizes the GP700 module options and describes the front and rear panels of the device.

Chapter 3, "Front-Panel Operation", explains how to operate the GP700 using the front-panel keypad and display.

Chapter 4, "Mini-Programs", explains how to automate simple tasks with GP700 mini-programs using the front-panel keypad and display.

Chapter 5, "Remote Operation", contains a short introduction to GPIB and RS-232 interfacing, followed by interface-specific configuration information. The chapter ends with a discussion of programming and controlling mini-programs remotely

Chapter 6, "Remote Commands", summarizes the categories of remote commands and defines the syntax and application of each command.

Chapter 7, "Servicing and Troubleshooting", suggests solutions to common problems, lists remote error codes, and gives instructions for contacting DiCon.

Appendix A, "Specifications", lists the optical, electrical, and mechanical specifications of the GP700.

Appendix B, "Modules", discusses the five module types and explains how each type is represented in your GP700 configuration diagram.

Appendix C, "Sample Configurations", presents and explains sample GP700 configuration diagrams.

Appendix D, "Device Compatibility", suggests strategies for porting GPIB and RS-232 code originally developed for previous generations of DiCon programmable switches.

Conventions

The names of remote commands and front-panel keys are in **BOLD**. Examples and fragments of remote code are in `constant width`. Command parameters for which you must substitute specific values are in *italic*. In command strings, parameter names are enclosed in angle brackets ("<>"). Optional parameters are enclosed in braces ("{}"). Data values are in `constant width` within proportional-width quotation marks.

Chapter 1
Unpacking and Setting Up

This chapter contains information about:

- unpacking the GP700
- fiberoptic cable and connector care
- mounting guidelines
- applying power to the GP700

Unpacking

Caution

Improper handling of the GP700 can damage optical cables and connectors. Do not pull or tug on fiber cables. Do not allow fiber cables to bend more tightly than a radius of 35 mm. Excessive bending or straining of these cables will break the optical fiber.

The following standard accessories are shipped with the GP700:

- GP700 Operation Manual
- fiberoptic handling guide
- power cord (NEMA 5-15P plug standard, European CEE 7-VII plug by request)
- front handles (4U and 6U rackmount chassis)
- hex tool (4U and 6U rackmount chassis)
- final test report
- configuration diagram.

Carefully remove the GP700 and accessories from the shipping container. Inspect the contents for any evidence of shipping damage. In particular, check for bent or damaged connectors. If your GP700 is configured with pigtailed fiber, you may find a few short lengths of bare, broken fiber in the packing container. This is not unusual, and does not necessarily mean the unit has been damaged.

Finally, initiate a self-test by connecting the GP700 to a power source (see "Applying Power", page 3) and pressing the front-panel rocker switch to the on position (|). The GP700 performs a self-test on power-up. If an error is found, the front-panel ERR indicator lights. Determine the nature of the error by pressing the **HELP** key.

Notify DiCon Fiberoptics immediately if the package contents are incomplete, if there is any sign of damage, or if the device does not pass the self-test. Please retain all packing materials for use in the event that the unit must be returned for servicing.

Handling Fiberoptic Cables and Connectors

Your instrument may come with fiber pigtail outputs. Treat cables with care to avoid cable damage and minimize optical loss. The minimum bend radius for most optical cables is 35mm. Bending optical cable more sharply than this specification may break the fiber or degrade optical performance.

Caution

Improper handling of the optical connectors can permanently damage the GP700. Proper handling consists of proper storage of the connectors, proper cleaning of the connectors, proper mating of the connectors, and proper handling of fiberoptic cables.

- Avoid bending the optical cable near a cable strain relief boot. Bending an optical cable near a strain relief boot is an easy way to permanently damage the optical fiber.
- Avoid bending the optical cable over a sharp edge.
- Avoid using cable tie wraps to hold optical cable. Overly tight tie wraps can create microbends or break an optical cable. Microbends can cause a dramatic reduction in optical performance.
- Do not pull on the bare fiber as this can break the fiber inside the component.
- Avoid using soldering irons near optical cable. Accidental damage can easily occur when a soldering iron is used near an optical cable. In addition, solder splatter can contaminate and permanently damage optical connectors.
- In order to obtain the most stable, repeatable optical performance, immobilize optical cables using wide pieces of tape or some form of mechanical cushion after the optical cables have been connected.

Storing Optical Connectors

All switches are shipped with dust caps in place covering all optical connectors. To prevent damage from dust contamination and other hazards, optical connectors should remain covered at all times when the instrument is not in use.

Cleaning Optical Connectors

Clean any exposed connectors using a cleaning kit supplied by the connector manufacturer or high-grade isopropyl alcohol and a cotton swab. To clean with alcohol and a swab, dab the tip of a cotton swab in alcohol and then shake off any excess alcohol. The tip should be moist, *not* dripping wet. Stroke the swab tip gently across the surface of the connector and around the connector ferrule. Allow the connector a minute to dry, or blow dry the connector using compressed air. Be careful when using compressed air because improper use may deposit a spray residue.

Mating Optical Connectors

- Clean both connectors prior to mating. Any small particles trapped during the mating process can permanently damage the connector.
- Insert the appropriate connector ferrule into the adapter smoothly. Do not allow the fiber tip to contact any surface. If the tip accidentally contacts a surface before mating, *stop*. Re-clean the connector and try again.
- Tighten the connector until it is finger tight, or to the torque specified by the connector manufacturer. Do not over-tighten the connector as this can lead to optical loss and connector damage.
- Check the optical insertion loss. If the loss is unacceptable, remove the connector, reclean both ends of the mate, and reconnect. You may have to repeat this process several times before a low-loss connection is made.
- After you make the connection, monitor the stability of the optical throughput for a few minutes. Optical power trending (slowly increasing or decreasing) is caused by the slow evaporation of alcohol trapped in the connection. Continue to monitor optical power until it stabilizes. If the loss is unacceptable, reclean the connectors and start again.

Rackmount Guidelines

When mounting the GP700 in a rack or electronic cabinet, follow the rack supplier's mounting instructions and use hardware rated for the weight of the device. The GP700 Rackmount Chassis conforms to IEC 297-1 and DIN 41494 Part I standards.

- If you intend to use the supplied handles, attach the handles with the supplied tool kit prior to mounting the GP700 in the rack.
- The front handles supplied with your GP700 cannot support the entire weight of the device. Do not lift the GP700 by the handles only.
- Use supplier-recommended hardware such as shelves or brackets to support the GP700. Do not mount the GP700 by the front flanges only.

Applying Power

AC Power Requirements

To operate the GP700 with AC power, connect the GP700 to a power source meeting the requirements given in Table 1-1. See Appendix A, "Specifications", for more information.

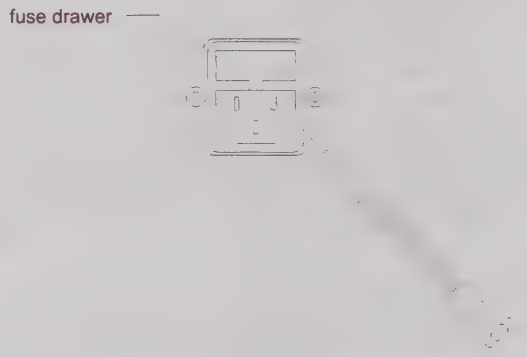
Table 1-1: *A/C Power Requirements*

Characteristic	Requirement
Input Voltage	100 - 240 VAC
Frequency	50 - 60 Hz
Power	150 VA max.

AC Fuse Requirements

The GP700 comes with a fused power inlet. If your GP700 is configured for AC operation, the fuse is stored within the power inlet (see Figure 1-1). To check the fuse, first disconnect the GP700 from line power and remove the power cable. Next, gently pry the fuse drawer free of the power inlet using the tip of a flat-blade screwdriver. If the fuse is defective or missing, replace it with a new fuse. DiCon recommends a 2A, 250V, fast-blow fuse.[1]

Figure 1-1: *AC Power Inlet Fuse Drawer*



AC Power Cord

The GP700 is shipped with a standard power cord equipped with a three-prong North American plug (NEMA 5-15P). DiCon may substitute a European cord (CEE 7-VII) by customer request. Customers from different regions may require a different plug style, and must provide their own power cord appropriate for their location and power source. The power inlet is a standard IEC 320 C14 receptacle.

Warning

Failure to properly ground the GP700 can result in personal injury. Before applying power, you MUST connect the protective earth ground of the power cable to the protective earth ground contact of the power source. DO NOT attempt to bypass or defeat this safety connection.

1. Use a 2.5-A fuse for 6U-rackmount matrix switches.

Chapter 2
Introduction

This chapter contains information about:

- the GP700 user interface
- module status indicators
- rear-panel configuration

Product Overview

The GP700 serves as a platform for the integration of multiple optical attenuator, filter, and switch modules. The device accommodates a nearly infinite range of configurations using a common interface, greatly reducing training, programming, and integration time when multiple GP700's are used. Table 2-1 summarizes the five basic module types used in a GP700.

Table 2-1: *GP700 Module Types*

Module Type	Description
A	variable attenuator
F	tunable filter
I	matrix switch
M	multi-channel switch
S	two-position switch

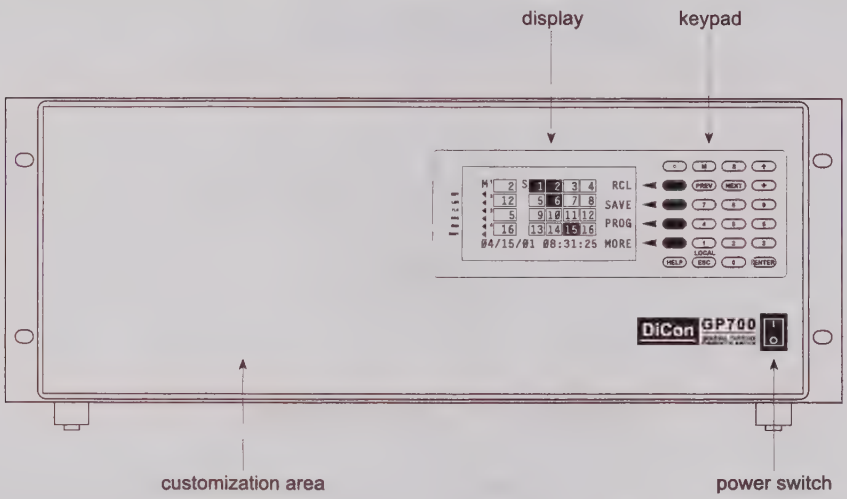
Attenuator modules typically set attenuation levels in the range 0–60dB with high resolution. Filter modules typically tune the center wavelength of a narrow passband over a 30-nm range. Matrix switch modules move each of *M* input fibers into alignment with each of *N* output fibers. Multi-channel switch modules move one, two, or three common fibers into alignment with one, two, or three of several output fibers. Two-position switch modules are On-Off, 1×2, and 2×2 fiberoptic switches. For more information on GP700 component modules, refer to Appendix B, "Modules".

In addition to serving as a platform for controlling independent modules, the modules themselves may be configured with internal optical connections. It is also possible to integrate any of a number of non-mechanical optical elements (isolators, couplers, etc.). Consult your configuration diagram whenever you have a question about the configuration of your device. For sample configurations, refer to Appendix C, "Sample Configurations".

The Front Panel

The front panel of the GP700 is shown in Figure 2-1. The front panel consists of a display, a keypad, a power switch, and a customization area which may or may not contain optical connectors.

Figure 2-1: *GP700 Front Panel (4U Rackmount Chassis)*



The Keypad

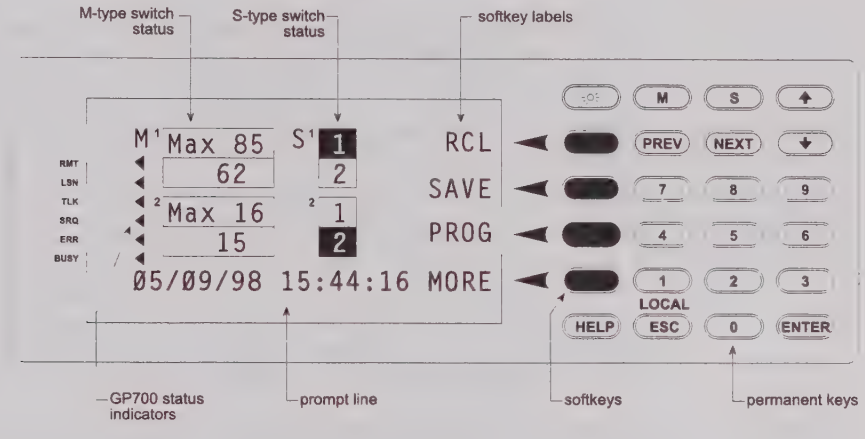
The keypad consists of twenty permanently labeled keys, and four blue softkeys. The functions of the four blue softkeys are determined by the corresponding softkey labels in the display. The softkey labels may change during data entry from the keypad. The functions of the permanent keys are given in Table 3-1. The functions of the main-menu softkeys are given in Table 3-2.

The Display

The GP700 front-panel display shows the current status of the component modules of the GP700. The status of each module appears in one or more status boxes. If your display cannot accommodate all modules on a single screen, your keypad will have a ⧉ key. Use the ⧉ key to cycle through the status screens. Not all GP700s have a ⧉ key. The exact look of the display depends on your particular configuration.

Figure 2-2 shows the relative placement of display elements for a GP700 configured with two multi-channel switch modules and two two-position switch modules. The status indicators are lit for the purpose of illustration.

Figure 2-2: *GP700 Front-Panel Display*



Two-Position Switch Module Status

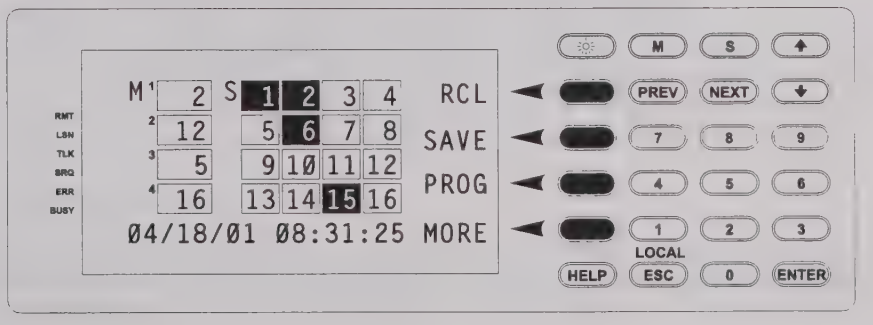
The status boxes of all two-position switch modules are grouped and labeled with a single "S". There are two types of status boxes for S-type modules.

The front-panel does not differentiate between the various S-type module configurations. Rather, the GP700 reports status in terms of two generic states. Depending on configuration, S-type modules in state 1 may be set to off, channel 1, or bypass position. Modules in state 2 may be set to on, channel 2, or insert position. The actuation style of 2×2 switch modules may vary from device to device. Refer to your configuration diagram to verify the actuation style of your modules.

Each S-type module may be represented by a single box containing the module number. If the box *is not* highlighted, the module is set to state 1. If the box *is* highlighted, the module is set to state 2. Figure 2-3 shows the display of a GP700 configured with four M-type modules and sixteen S-type modules. Modules S1, S2, S6, and S15 are set to state 2. The remaining S-type modules are set to state 1.

Depending on the configuration of your device, each module may be represented by two boxes with the numbers "1" and "2" indicating the state of the module. The module number appears to the upper left side of the boxes. If state "1" is highlighted, the module is set to state 1. If state "2" is highlighted, the module is set to state 2. In Figure 2-2, S-type module S1 is set to state 1 and module S2 is set to state 2.

Figure 2-3: *Front-Panel Display With Multiple M- and S-Type Modules.*



Multi-Channel Switch Module Status

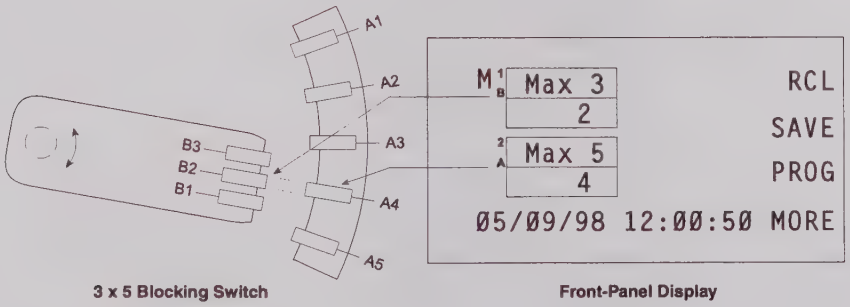
The status boxes of all multi-channel switch modules are grouped and labeled with a single "M". The status of each module may appear in one, two, or four boxes.

Each M-type module may be represented by a single box containing the current channel setting. In this case, the module number appears to the upper left side of the box. The four M-type modules in Figure 2-3 are displayed in one-box format. Module M1 is set to channel 2; M2 is set to channel 12; M3 is set to channel 5; and M4 is set to channel 16.

Depending on the configuration of your device, each M-type module may be represented by two boxes. The module number appears to the upper left side of the boxes. The top box indicates the maximum output channel. The bottom box indicates the current channel setting. The two M-type modules in Figure 2-2 are displayed in two-box format. Module M1 has 85 output channels and is currently set to channel 62. Module M2 had 16 output channels and is currently set to channel 15.

For M-type modules with multiple input settings, the GP700 displays both input and output settings. The input setting is labeled port "B". The output setting is labeled port "A". Figure 2-4 shows an example of the display of a GP700 configured with a single 3×5 M-type module, as well as a schematic of the actual fiber alignment. Port B is set to input channel 2. Port A is set to output channel 4.

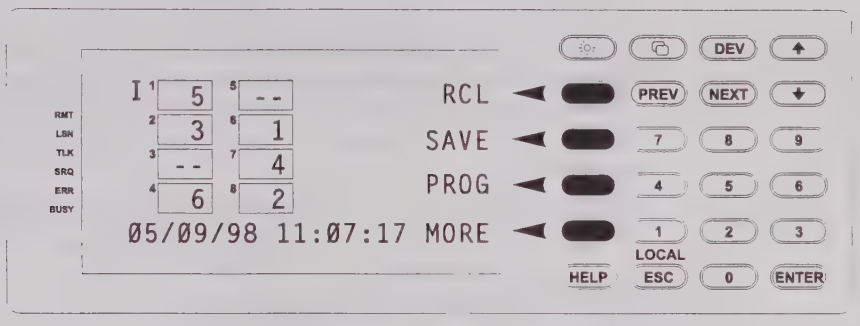
Figure 2-4: *Front-Panel Display With 3×5 M-Type Module*



Matrix Switch Module Status

Matrix switch module status is represented by a group of boxes, one box for each matrix input. The input channel number appears at the upper left corner of each box. The current output channel connection appears inside each box. An output channel value of "- -" indicates that there is no optical connection. Figure 2-5 shows the display of a GP700 configured with a single 8×6 I-type module. Input channels 3 and 5 are set to "no connect". Input channels 1, 2, 4, 6, 7, and 8 are connected to output channels 5, 3, 6, 1, 4, and 2, respectively.

Figure 2-5: *Front-Panel Display With I-Type Module*

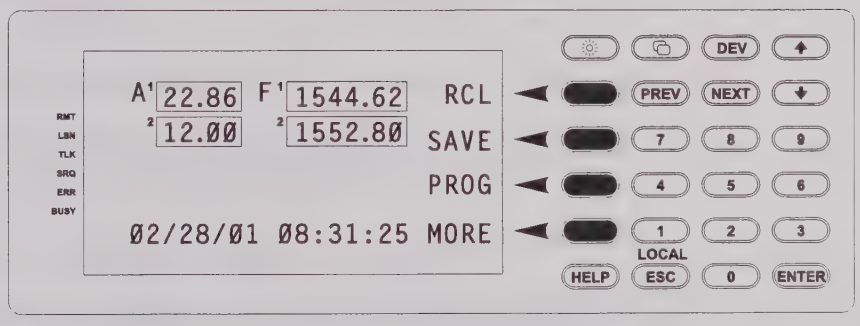


Attenuator and Filter Module Status

The status boxes of all attenuator modules are grouped and labeled with a single "A". All filter modules are grouped and labeled with a single "F". The status of each module appears in a single box. Each A-type module status box contains the current attenuation level in dB. Each F-type module status box contains the current center-wavelength setting in nm. The module number appears to the upper left side of each box.

Figure 2-6 shows the front-panel display of a GP700 configured with two A-type modules and two F-type modules. The two A-type modules are set to 22.86dB and 12.00dB. The two F-type modules are set to 1544.62nm and 1552.80nm.

Figure 2-6: *Front-Panel Display With A- and F-Type Modules*



Status Indicators

The left side of the front-panel display contains six triangular status indicators that light when a corresponding status condition exists. The indicators are unlit when the condition does not exist. The name of each indicator is printed on the front panel next to the display. The six status conditions are: busy (BUSY), error (ERR), remote interface (RMT), talk (TLK), listen (LSN), and service request (SRQ).

Busy Status Indicator

The BUSY indicator lights whenever a module action is taking place. Front-panel keypad entry is buffered while the GP700 is busy. The buffered keystrokes are executed when the busy condition ends.

Error Status Indicator

The ERR indicator lights when the GP700 encounters an error during keyboard entry or remote operation, including during the recording and running of mini-programs. If the error is generated locally, pressing **HELP** will display a short explanation and clear the error. Pressing any other key will clear the error without displaying any message. The help screen saves only one error message at a time. It is not necessary to clear the error before continuing.

It may be necessary to return the GP700 to local mode, either remotely or by pressing the **LOCAL** key, before you are able to use the **HELP** function from the front panel.

Remote Interface Status Indicator

The RMT indicator lights whenever the GP700 enters remote mode. This occurs whenever the GP700 is addressed via the GPIB interface. While in remote mode, the GP700 ignores most keypad entry. To return the GP700 to local mode (unlock it), press the **LOCAL** key or send the GPIB Enable Local message remotely (see "Local, Remote, and Local Lockout Modes", page 39).

Talk Interface Status Indicator

The TLK indicator lights whenever information is placed in the GP700 GPIB output queue in response to a GPIB query command. The indicator turns off after the information is read from the queue.

Listen Interface Status Indicator

The LSN indicator lights whenever information is placed in the GP700 input queue via the GPIB interface. The indicator remains lit until the GP700 receives a command terminator.

Service Request Interface Status Indicator

The SRQ indicator lights whenever the GP700 sends a GPIB service request (see "Command Timing", page 39 and ***SRE**, page 71).

Softkey Labels

The softkey labels describe the function of the blue buttons located to the immediate right of the display. To access a softkey function press the corresponding button.

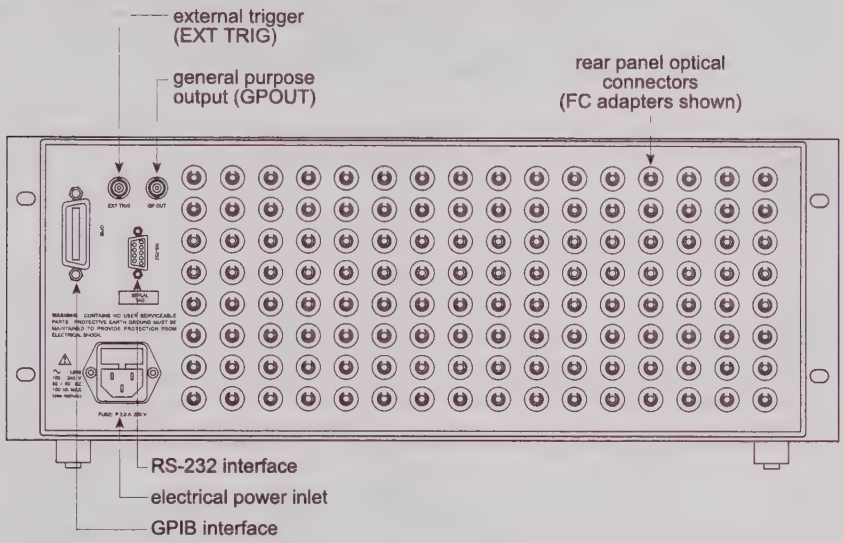
The Prompt Line

The prompt line conveys messages to you when you are using the keypad. While the GP700 is in the idle state, the prompt line displays the current date and time.

The Rear Panel

The rear panel of the GP700 is shown in Figure 2-7. The rear panel contains the GPIB connector, the RS-232 connector, the external trigger input connector, the general purpose output connector, the power inlet, and a customization area which may or may not contain optical adapters and/or pigtails.

Figure 2-7: *Rear Panel of the GP700 (Rackmount Chassis)*



GPIB Receptacle

The GPIB receptacle provides access to the GP700 remote GPIB interface. The receptacle is a standard IEEE488 GPIB receptacle.

RS-232 Receptacle

The RS-232 receptacle provides access to the GP700 remote RS-232 interface. The receptacle is a standard 9-pin D-subminiature female receptacle.

External Trigger Receptacle

The external trigger receptacle provides access to the GP700 external trigger input. External triggers are meaningful only in the context of mini-programs. The receptacle, a standard BNC bulkhead connector, is labeled EXT TRIG.

General Purpose Output Receptacle

The general purpose output receptacle provides access to the GP700 general purpose output. The receptacle, a standard BNC bulkhead connector, is labeled GP OUT.

Optical Adapters and Feedthroughs

Optical adapters and feedthroughs provide access to the optical inputs and outputs of the component modules. The exact form of the ports depends on the configuration purchased. Figures 2-8 and 2-9 show some common connectors, adapters, and pigtail feedthroughs.

The marking of each rear panel is unique, and depends upon the device configuration. In general, the rear panel is marked to indicate module inputs and outputs. Where there are internal connections, some module inputs and outputs may not be available from the rear panel. For input and output naming conventions, see Appendix B, "Modules".

Figure 2-8: *Pigtail Feedthroughs*

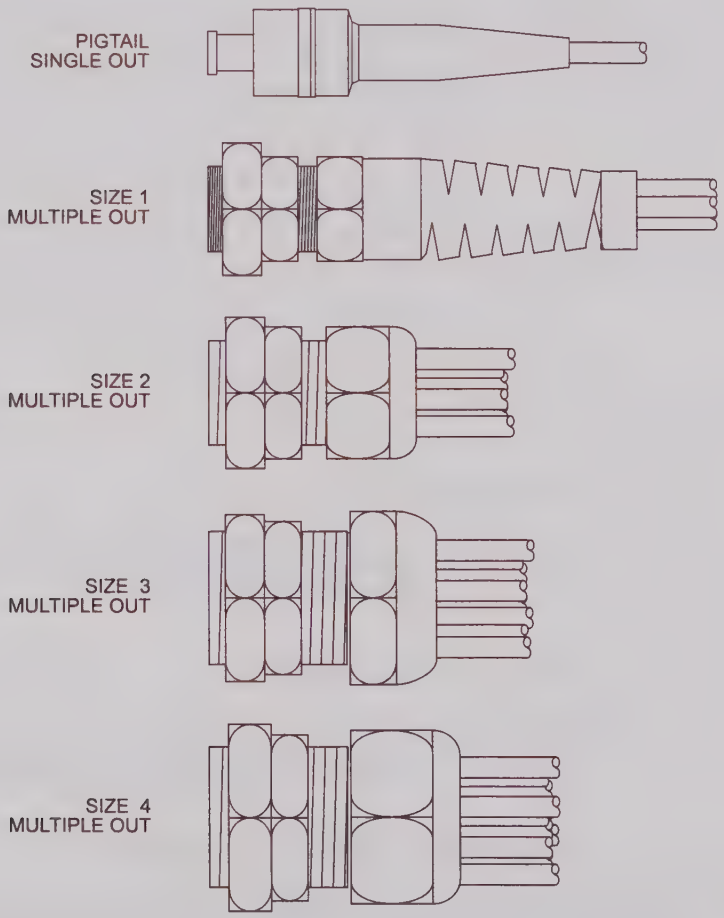
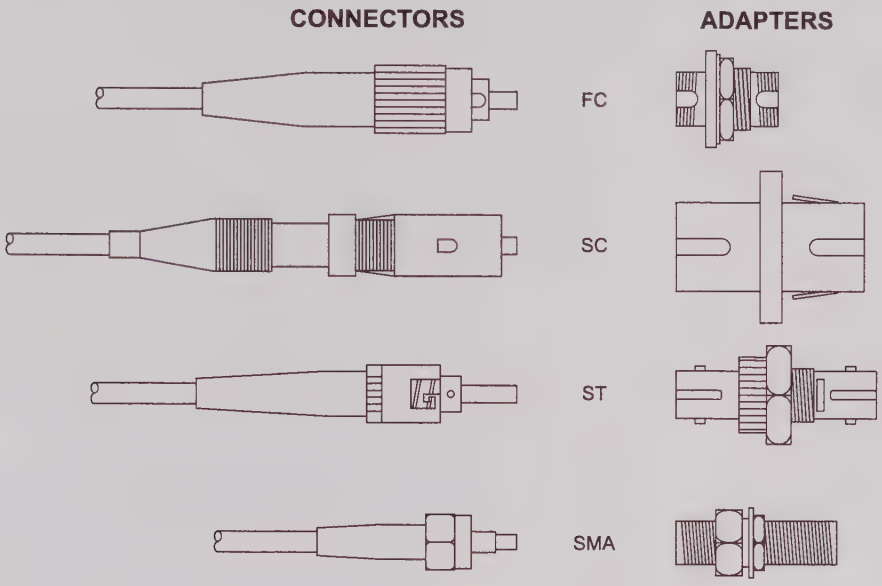


Figure 2-9: *Connectors and Adapters*



Chapter 3
Front-Panel Operation

This chapter contains information about operating the GP700 using the front-panel display and keyboard.

Starting the GP700

Caution

Read Chapter 1, "Unpacking and Setting Up", before connecting optical connectors or applying power. Improper handling of fiberoptic cables and connectors can permanently damage the GP700.

Turn on the GP700 by pressing the front-panel rocker switch to the on position (|). Upon applying power, the display shows the DiCon logo, the model number of the switch, and the firmware version. Next, the GP700 performs a self-test and returns all component modules to reset position. The reset states of each module type are defined in Table 6-6, "Soft Reset State", on page 67.

If the GP700 display fails to light, set the rocker switch to the off position (○), check all power connections, and return the rocker switch to the on position (|). Contact DiCon Fiberoptics if the problem persists.

Operating the GP700

When all pending operations are complete and no modules are selected (no blinking module icons on the front-panel display), the GP700 is said to be in the idle state. While in idle state there are no active modules and the prompt line displays the current time and date. The idle state is also called the "main menu" because the softkeys are set to their top-level functions. Under most conditions, you can return the GP700 to the idle state by pressing the **ESC** key three times in a row. The keypad permanent functions and main-menu softkey functions are given in Tables 3-1 and 3-2.

Table 3-1: Main-Menu Permanent Keys

Key	Function
☼	Adjusts display brightness.
M	Activates the last M-type module changed. Repeatedly pressing the **M** key cycles through the available M-type modules. For devices without an **M** key, use the ▢ and **DEV** keys instead.
S	Activates the last S-type module changed. Repeatedly pressing the **S** key cycles through the available S-type modules. For devices without an **S** key, use the ▢ and **DEV** keys instead.
▢	Advances the display to the next screen. The ▢ key does not appear on some devices.
DEV	Activates device selection mode. Repeatedly pressing the **DEV** key cycles through the module types available on the current screen. For devices without a **DEV** key, use the **S** and **M** keys instead.
↑	Increments the setting of the active module. Toggles the setting of a two-position switch module. Matrix switch modules move to position only after you press the **ENTER** key. All other modules move immediately.
↓	Decrements the setting of the active module. Toggles the setting of a two-position switch module. Matrix switch modules move to position only after you press the **ENTER** key. All other modules move immediately.
NEXT	Activates the next module.
PREV	Activates the previous module.
0 – 9	Used for direct entry of data.
HELP	Provides context-sensitive instructions.
ESC	Backspaces or cancels an instruction, or restores local control when in remote mode.
ENTER	Completes and confirms data-entry operation.

Table 3-2: Main-Menu Softkeys

Key	Function
RCL	Recalls a saved device state.
SAVE	Saves a complete or partial device state.
PROG	Enables entry or execution of GP700 mini-programs.
MORE	Shows additional softkeys.
I/O	Configures the external trigger port and general purpose output port.
GPIB	Configures the GPIB interface address.
232	Configures the RS-232 interface baud rate.
CLCK	Sets the internal clock time and date.

Getting Help

To obtain information on the function of a particular key, press the **HELP** key followed by the key about which you want information. To obtain information about a softkey, first press the key sequence necessary to see the softkey on the display, then press the **HELP** key followed by the softkey about which you want information. The GP700 responds with a short description of the key function. This information is intended as a reminder of the key function, not as a substitute to reading this manual. Press **HELP** again to get out of the help function.

You can also use the **HELP** key to get more information when an error occurs. When the ERR indicator lights, press **HELP** to get the error code and explanation. For more information about the cause of the error see Table 7-2, "Remote Error Codes", on page 83.

Returning to the Main Menu

Except when you are recording or running a GP700 mini-program, you can always return to the main menu (the "idle state") by repeatedly pressing the **ESC** key. While in mini-program record mode, press **PROG** followed by **EXIT** to exit record mode and return to the main menu. While in mini-program run mode, press **ABRT** and wait a few seconds to return to the main menu.

When you are in GPIB remote mode, pressing **ESC** returns the keypad to local mode (see "Local, Remote, and Local Lockout Modes", page 39).

Changing Display Brightness

Press the ☼ key to change the brightness of the display. Each time you press the ☼ key the brightness level decreases by 25%, until the display finally goes dark. To return the display to 100% brightness press ☼ one more time.

Changing Module Settings

To change a module setting, first activate the desired module, then change the module setting. You can activate a module by pressing one of the module activation keys: **M**, **S**, or **DEV**.

Depending on the configuration of your device, your keypad has either an **M** key and an **S** key or a **DEV** key and a ⧉ key. For devices configured with **M** and **S** keys, use the **M** key to activate M-type modules and the **S** key to activate S-type modules. For devices configured with **DEV** and ⧉ keys, use the **DEV** key to activate any module. You may have to press the ⧉ key if all modules do not fit on a single display screen.

After pressing an activation key, the active module box blinks. The GP700 displays the active module number and its current channel, attenuation, or center-wavelength setting in the prompt line. To select a different module of the same type, press **NEXT** or **PREV** to cycle through the available choices.

Press ↑ or ↓ to increment, decrement, or toggle the channel, attenuation, or center-wavelength setting of the active module. Matrix switch modules move to position only after you press the **ENTER** key. All other modules move immediately. To change the module setting directly, enter the new value using the numeric keypad, then press **ENTER**. The BUSY front-panel indicator lights while the module changes setting. Any keyboard entries made while the device is busy are buffered, then executed as soon as the device is no longer busy.

Finally, press **ESC** to deselect the active module and return to the idle state to avoid inadvertently changing the current setting.

Saving a Device State

The GP700 has nine registers that you can use to save the state of the device. The device state can be saved as a complete or partial device state. A complete device state includes the states of all component modules. A partial device state includes the states of a subset of component modules.

Saving a Complete Device State

To save the complete device state press the **SAVE** softkey. You may have to press **MORE** a few times to see the **SAVE** key. The GP700 displays the number of the last-saved register in the prompt line. Increment or decrement the register number using the ↑ and ↓ keys, or directly enter the desired register number using the numeric keypad. Once you've selected the desired register press **ENTER** to save the state. Upon saving, the previous contents of the selected register are overwritten without warning. To cancel without saving state press **ESC**.

Saving a Partial Device State

To save a partial device state press the **SAVE** softkey. You may have to press **MORE** a few times to see the **SAVE** softkey. Press the **EDIT** softkey to deselect modules. The GP700 assumes that you wish to save the states of all modules. You must deselect a module to prevent the module state from being saved.

To deselect a module, press **M**, **S**, or **DEV** to choose the module type. You may have to press the ⧉ key to access modules on a different screen. Press **NEXT** or **PREV** to cycle through the available modules. The GP700 displays the module currently available for selection or deselection in the prompt line. When you've reached the desired module press the **DEL** softkey to deselect a module. Press the **ADD** softkey to select a deselected module. Alternatively, you can press the ↑ and ↓ keys to toggle the module's selection status. All deselected modules *blink*. Continue cycling through modules and toggling selection status until you are happy with the results.

When you are satisfied with the selection status of all modules, press the **SAVE** softkey. The GP700 displays the number of the last-saved register in the prompt line. Increment or decrement the register number using the ↑ and ↓ keys, or directly enter the desired register number using the numeric keypad. Once you've selected the desired register press **ENTER** to save the state. Upon saving, the previous contents of the selected register are overwritten without warning. To cancel without saving the state press **ESC**. To exit save mode press **ESC** repeatedly until you return to the idle state.

Recalling a Device State

Press the **RCL** softkey to recall a partial or complete device state. You may have to press **MORE** a few times in order to see the **RCL** softkey. The GP700 displays the number of the last-recalled register in the prompt line. Increment or decrement the register number using the ↑ and ↓ keys, or directly enter the register number using the numeric keypad. Once you've selected the desired register press **ENTER** to recall the device state. To cancel without recalling the device state press **ESC**.

When you recall a partial device state, only the modules that were selected when the state was saved are returned to their saved settings; the unsaved modules remain at their current settings.

Recording a Mini-Program

To record a mini-program press the **PROG** softkey followed by **REC**. Next, enter the mini-program commands. To exit record mode without saving the program, press **PROG** followed by **EXIT**. To end recording and save a mini-program, press **PROG** followed by **SAVE**. Increment or decrement the program number using the ↑ and ↓ keys, or directly enter the desired program number (0–3) using the numeric keypad. You may save up to four mini-programs. Press **ESC** to return to record mode without saving. Press **ENTER** to save the program. Upon saving, the previous contents of the selected program are overwritten without warning. Recording mini-programs from the front panel is described in detail in Chapter 4, "Mini-Programs".

Running a Mini-Program

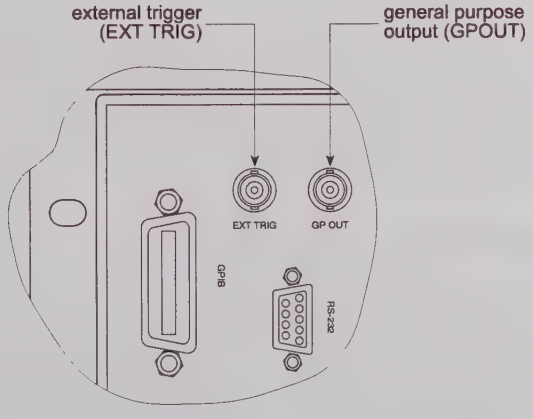
To run a mini-program, press the **PROG** softkey then **RUN**. The GP700 displays the number of the last-run mini-program in the prompt line. Increment or decrement the program number using the ↑ and ↓ keys, or directly enter the desired program number using the numeric keypad. Press **ENTER** to begin execution.

To stop the currently running mini-program, press the **ABRT** softkey. After a few seconds the GP700 will return to the main menu. If the mini-program was executed remotely the GP700 may be in remote mode, and the keyboard may be locked out. In this case, press the **LOCAL** key to unlock the keyboard and return it to local mode, then press **ABRT** to stop the mini-program.

Configuring the Input/Output Ports

The GP700 is equipped with two TTL I/O ports located on the rear panel. A detail of the rear panel is shown in Figure 3-1.

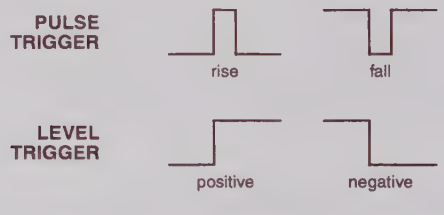
Figure 3-1: *GP700 I/O Ports*



The external trigger input port (EXT TRIG) can receive a mini-program trigger (see "Waiting for a Trigger", page 27). The general purpose output port (GP OUT) can output a trigger generated by a mini-program (see "Sending a General Purpose Output", page 28) or by a remote command (see **GPOUT**, page 58).

There are two types of triggers: pulse triggers and level triggers. Pulse triggers can be set to rising-edge or falling-edge polarity. Level triggers can be set to positive or negative polarity. The GP700 trigger types and polarities are illustrated in Figure 3-2.

Figure 3-2: *I/O Port Trigger Signals*



To configure the I/O ports, press the **I/O** softkey. You may have to press **MORE** a few times in order to see the **I/O** key. The GP700 displays the last-configured port in the prompt line. Position the blinking cursor beneath the entry you wish to change using the ← and → softkeys. Change the entry to the desired value with the topmost softkey. Press **ENTER** to store the port configuration, or **ESC** to return to the idle state without saving.

Changing the GPIB Address

Press the **GPIB** softkey to change the primary GPIB address. You may have to press **MORE** a few times in order to see the **GPIB** key. The GP700 displays the current GPIB address in the prompt line. Enter the new address using the numeric keypad or the ↑ and ↓ keys. Integers from 1 to 15 are valid GPIB addresses. Depending on your hardware configuration, integers from 16 to 30 may also be valid addresses. Before changing the address, first verify it will not conflict with any other instruments that share the GPIB bus. Press **ENTER** to save the new address, or **ESC** to return to the idle state without saving. The GPIB address is factory-set to 3.

Changing the RS-232 Baud Rate

Press the **232** softkey to change the RS-232 baud rate. You may have to press **MORE** a few times in order to see the **232** key. The GP700 displays the current baud rate in the prompt line. Select the new baud rate with the ↑ and ↓ keys. You may select either 1200 or 9600 baud. Press **ENTER** to save the new baud rate, or **ESC** to return to the idle state without saving. The RS-232 baud rate is factory set to 9600.

Setting the Clock

Press the **CLCK** softkey to set the time and date. Change the time by pressing the **TIME** key, or the date by pressing the **DATE** key. Position the blinking cursor in the prompt line beneath the entry you wish to change using the ← and → softkeys. Change the entry to the desired value using the ↑ and ↓ keys. Press **ENTER** to store the new clock value, or **ESC** to return to the idle state without saving.

Chapter 4
Mini-Programs

This chapter describes entering and running mini-programs from the front panel. For information about entering and running mini-programs remotely, see "Implementing Mini-Programs Remotely", page 42.

Overview

Mini-programs can be a time-saving tool for automating basic tasks. You can record, store, and run GP700 mini-programs directly from the front panel, or if you prefer, via the remote interfaces. To use mini-programs you need neither programming experience nor a computer or external controller. All you need is a basic understanding of GP700 front-panel operation.

The GP700 supports a limited instruction set of key commands which allow you to automate typical testing tasks. The command set allows you to:

- Change module settings
- Recall device states
- Wait for trigger inputs
- Send trigger outputs

Examples of possible mini-program applications include:

- Setting up standard test matrices where each step in the test, regardless of the number of program commands in any given step, is executed by a single keystroke.
- Using the GP700 as a safety unit, where the device changes to a different state when it receives an external or remote trigger.
- Using the GP700 as a controller, where after achieving a given state the GP700 sends a trigger signal to other instruments in your experiment.

Mini-Program Example

The following example demonstrates some of the GP700's mini-program capabilities. The example is intended to introduce you to the kinds of tasks you can automate using mini-programs. The command set is fully described later in this chapter, so don't worry if you don't understand all of the steps.

The example mini-program is based on the test set-up shown in Figure 4-1. The purpose of the test is to measure the optical power passing through a reference and through fifteen devices at two different source wavelengths (1300nm and 1550nm). The test employs a GP700 configured with two 1×16 M-type modules (M1 and M2), and a single 1×2 S-type module (S1).

Figure 4-1: *Example Test Set-Up*

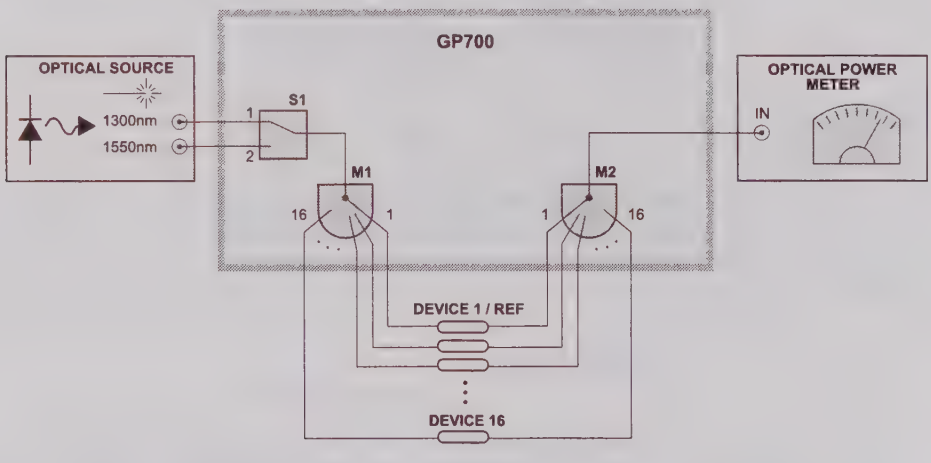


Table 4-1 lists the initial steps required to prepare the GP700 for our test mini-program. Table 4-2 lists the steps required to enter the mini-program.

Table 4-1: *Sample Program Initialization Sequence*

Keystrokes	Description
RCL **RST**	Reset the GP700.
M **1** **ENTER** **NEXT** **1** **ENTER**	Set modules M1 and M2 to the reference channel (channel 1).
S **1** **ENTER**	Set module S1 to channel 1, placing the 1300-nm source on-line.
SAVE **1**	Save the initial device state in register one.
RCL **RST**	Reset the GP700.

Table 4-2: Sample Program Entry Sequence

Keystrokes	Prompt Line	Description
PROG **REC**		Enter mini-program record mode.
RCL **1** ENTER	001>>RCL STATE:1	Set the test to its initial state (reference channel, 1300-nm source).
MORE **LOOP** **BGN** **15** **ENTER**	002>>BLOOP 15	Begin a loop for running the test on 15 devices under test and the reference channel.
WAIT **KEY** **ENTER**	003>>WAIT K	Pause the mini-program and wait for the user to press **ENTER**. This pause allows the user to note the 1300-nm results.
S ↑ **ENTER**	004>>S01:2	Toggle S1 to channel 2, placing the 1550-nm source on-line.
MORE **WAIT** **KEY** **ENTER**	005>>WAIT K	Pause the mini-program and wait for the user to press ENTER. This pause allows the user to note the 1550-nm results.
M ↑ **ENTER**	006>>M1:2	Increment M1 to the next channel.
M **NEXT** ↑ **ENTER**	007>M2:2	Increment M2 to the next channel. The next device is now on-line.
S ↑ **ENTER**	008>>S01:1	Toggle S1 to channel 1, placing the 1300-nm source back on-line.
MORE **LOOP** **END** **ENTER**	009>>ELOOP	Repeat the loop until all fifteen devices are tested.
MORE **RCL** **1** **ENTER**	010>>RCL STATE:1	Clean up at the end of the test by returning the GP700 to the predefined initial state.
PROG **SAVE** **0** **ENTER**	>SAVE AS:0	Save mini-program zero and exit record mode.

Mini-programs can greatly reduce the number of keystrokes required to perform common tasks. Table 4-3 compares the number of keystrokes required to complete the sample test using standard front-panel operation against the number of keystrokes required when using the mini-program. For this comparison it is assumed that the program is stored as program 0, and that in both cases register 1 contains the initial set-up.

Table 4-3: Keystroke Comparison of Non-Automated Versus Automated Testing

Not Automated	Automated	Comment
RCL **0**	**PROG** **RUN** **0**	Go to device 1 and measure at 1330nm.
S ↑	**ENTER**	Measure at 1550nm
M ↑ **NEXT** ↑ **S** ↓	**ENTER**	Change to device 2. Measure at 1300nm.
S ↑	**ENTER**	Measure at 1550nm
M ↑ **NEXT** ↑ **S** ↓	**ENTER**	Change to device 3. Measure at 1330nm.
S ↑	**ENTER**	Measure at 1500 nm.
• • •	• • •	• • •
RCL **0**	**ENTER**	Return to initial state for next series of tests.

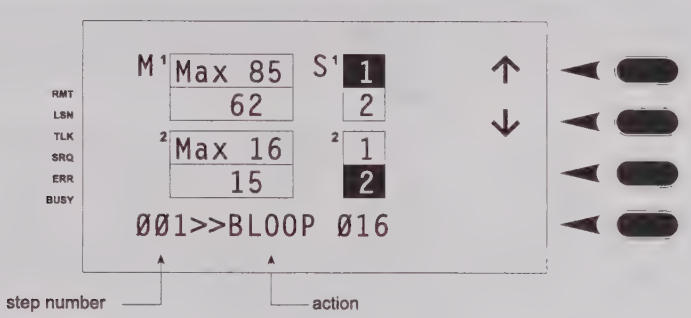
Without automation, the user must make several keystrokes to accomplish what takes only a single keystroke using the mini-program. Over the course of the test, the non-automated approach requires approximately 130 keystrokes while the automated approach requires 32 keystrokes (a 75% reduction). Not only does the mini-program save time, but it greatly reduces the potential for incorrect keystroke entry.

The following sections describe in detail how to record and run a mini-program. If you already know how to use the GP700 then you are only about a half-hour away from becoming a mini-programming whiz.

Recording a Mini-Program

To record a mini-program press the **PROG** softkey followed by **REC**. The GP700 display prompt indicates the current program step number. The instruction you are currently programming follows the prompt. A sample record-mode front-panel display is shown in Figure 4-2.

Figure 4-2: *Record-Mode Front-Panel Display*



It is possible to store approximately 100 steps in a single mini-program, depending on the memory requirements of each step and the requirements of the other stored mini-programs. If you enter too many program steps, the GP700 will warn that you risk corrupting the mini-program memory and destroying other stored mini-programs.

While in record mode, press **PROG** followed by **EXIT** to return to the main menu without saving any changes. Press **PROG** followed by **SAVE** to save the mini-program and return to the main menu. You cannot use **ESC** to return to the main menu while in record mode. To stop the currently running mini-program press the **ABRT** softkey. Program execution stops immediately, although it is possible that buffered commands will continue to execute.

Some keys behave differently in record mode than they do during normal operation. While in record mode, the following rules apply:

- Keys that cannot be recorded in a mini-program simply do not function.
- The **ESC** key cancels the current step. To exit a mini-program without saving changes you must press **PROG** followed by **EXIT**.
- The ↑ and ↓ keys increment or decrement a switch module setting relative to the current setting. A relative increment or decrement that results in an out-of-range position will generate an error.

The keys available while in record mode are given Table 4-3.

Table 4-4: *Mini-Program Record-Mode Key Functions*

Key	Function
M	Activates the last M-type module changed. Repeatedly pressing the **M** key cycles through the available M-type modules. For devices without an **M** key, use the ⧉ and **DEV** keys instead.
S	Activates the last S-type module changed. Repeatedly pressing the **S** key cycles through the available S-type modules. For devices without an **S** key, use the ⧉ and **DEV** keys instead.
⧉	Advances the display to the next screen. The ⧉ key does not appear on some devices.
DEV	Activates device selection mode. Repeatedly pressing the **DEV** key cycles through the module types available on the current screen. For devices without a **DEV** key, use the **S** and **M** keys instead.
NEXT	Activates the next module.
PREV	Activates the previous module.
↑	Increments the setting of the active module. Use the ↑ key to specify relative switch settings.

Table 4-4: Mini-Program Record-Mode Key Functions (Continued)

Key	Function
↓	Decrements the setting of the active module. Use the ↓ key to specify relative switch settings.
0 – 9	Used for direct entry of data.
RCL	Recalls a saved device state.
PROG	Accesses mini-program **SAVE** and **EXIT** softkeys.
MORE	Shows you more softkeys.
WAIT	Pauses execution of the mini-program, and waits for an trigger before continuing. The trigger may be generated from the keypad, via the external trigger input, or over the remote interfaces.
LOOP	Begins or ends a program loop.
GPO	Sends an output trigger.
HELP	Provides context sensitive instructions.
ESC	Backspaces or cancels currently action.
ENTER	Completes and confirms data-entry operation.

Changing a Module Setting

To change a module setting first activate the desired module, then change the module setting. Depending upon the configuration of your device, you can select a module by pressing one of the module selection keys: **M**, **S**, or **DEV**.

Depending on the configuration of your device, your keypad has either an **M** key and an **S** key or a **DEV** key and a ⧉ key. For devices configured with **M** and **S** keys, use the **M** key to activate M-type modules and the **S** key to activate S-type modules. For devices configured with **DEV** and ⧉ keys, use the **DEV** key to activate any module. You may have to press the ⧉ key if all modules do not fit on a single display screen.

The GP700 displays the active module number and its current channel, attenuation, or center-wavelength setting in the prompt line. To select a different module press **NEXT** or **PREV** to cycle through the available choices.

You can change module settings absolutely or relatively. *Absolute* settings specify the exact channel, attenuator, or center-wavelength setting for any given module. For example, changing an M-type module to channel 40, regardless of the current channel, is an absolute channel change. All attenuator and filter module settings are interpreted as absolute settings.

Enter absolute settings using the numeric keypad, then press the **ENTER** key. The active module display updates to the new setting, indicating the action has been recorded.

Relative changes increment or decrement the setting of a switch module, regardless of the current value at record time or at run time. For example, if an M-type module is currently set to channel 2, incrementing the setting by three channels will take it to channel 5. If the same module were instead currently set to channel 11, incrementing the setting by the same amount will take it to channel 14. A relative increment or decrement that results in an out-of-range position will generate an error.

Enter relative changes by pressing ↑ and ↓ to increment and decrement, respectively. When you reach the desired relative change, press the **ENTER** key. The GP700 updates the active module display, indicating that the action has been recorded.

It is possible to set all M-type modules or all S-type modules at the same time (synchronously) using the All Call remote commands. This feature is available only when a mini-program is entered remotely (see **M0**, page 60, and **S0**, page 68), or when the GP700 is factory-set for synchronous operation.

Recalling a Device State

To recall a device state from within a mini-program, press the **RCL** softkey. You may have to press **MORE** a few times in order to see the **RCL** key. The GP700 displays the last-recalled register in the prompt line. Increment or decrement the register number using the ↑ and ↓ keys, or enter it directly using the numeric keypad. Press **ENTER** when you've selected the desired register. When you recall a partial device state, only the modules that were selected when the state was saved are returned to their saved settings; the unsaved modules remain at their current settings.

Waiting for a Trigger

Press the **WAIT** softkey to place a wait in a mini-program. You may have to press **MORE** a few times in order to see the **WAIT** key. Each of the four mini-program wait types responds to a different trigger type. The default wait type responds to keyboard triggers. To choose a different wait type press the appropriate softkey. When an executing program encounters a wait, the wait type is indicated in the prompt line. Table 4-5 describes the available wait options.

Table 4-5: *Mini-Program Wait Options*

Softkey	Description
KEY	Keyboard Wait: the mini-program pauses until the user presses **ENTER**. This feature can be used to pause your program while a user notes intermediate results.
EXT	External Trigger Wait: the mini-program pauses until it receives a trigger signal via the external trigger port. The trigger type and polarity must match the configuration of the external trigger port.
GPIB	GPIB Interface Wait: the mini-program pauses until it receives a Trigger command (see ***TRG**, page 78) via the GPIB interface or a GPIB Group Execute Trigger (**GET**) message. For more information about the **GET** message refer to your GPIB controller documentation.
232	RS-232 Interface Wait: the mini-program pauses until it receives a trigger signal via the RS-232 interface (see ***TRG**, page 78).

Setting Up a Program Loop

Mini-program loops begin with a Begin Loop instruction and end with an End Loop instruction. There must be an End Loop instruction for every Begin Loop instruction. When the loop begins, the loop counter is initialized with a loop-count value. Upon reaching the end loop instruction, the loop counter is decremented and compared to zero. The loop repeats if the loop counter is not equal to zero. Otherwise, the loop ends and the mini-program continues with the next instruction.

To begin a mini-program loop, press the **LOOP** softkey then **BGN**. You may have to press **MORE** a few times to see the **LOOP** key. Enter the desired loop-count value using the numeric keypad or the ↑ and ↓ keys. Press **ENTER** to complete the Begin Loop instruction.

To end a mini-program loop, press the **LOOP** softkey then **END**. Press **ENTER** to complete the End Loop instruction. To create an infinite loop, enter a loop count of zero ("000"). To stop an infinite loop, abort the mini-program by pressing the **ABRT** softkey or by sending the Abort Mini-Program remote command (**PROGram:ABORT**, page 64).

Sending a General Purpose Output

To generate a general purpose output trigger signal from within a mini-program, press the **GPO** softkey. You may have to press **MORE** a few times to see the **GPO** key. Press **ENTER** to complete the instruction. The general purpose output instruction sends a signal via the general purpose output port according to the run-time port configuration (see"Configuring the Input/Output Ports", page 19).

Saving a Mini-Program

To save a mini-program and exit mini-program record mode, press the **PROG** softkey then **SAVE**. You may have to press **MORE** in order to see the **PROG** key. Enter the desired program number using the numeric keypad or the ↑ and ↓ keys. You may save up to four mini-programs numbered 0–3. Press **ESC** to return to record mode without saving. Press **ENTER** to complete the save instruction. Upon saving, the previous contents of the selected mini-program are overwritten without warning.

Exiting Record Mode Without Saving

To exit record mode without saving a mini-program, press the **PROG** softkey then **EXIT**. You may have to press **MORE** in order to see the **PROG** key. The GP700 discards all unsaved entries when you exit without saving.

Running a Mini-Program

To run a mini-program from the front panel, press the **PROG** softkey then **RUN**. You may have to press **MORE** a few times to see the **PROG** key. Enter the desired program number using the numeric keypad or the ↑ and ↓ keys. Press **ENTER** to begin execution. The prompt line displays the current program number and a spinner indicating that the program is executing. It is not possible to run a mini-program within a mini-program.

To run a mini-program via a remote interface, use the Execute Mini-Program remote command (**PROGram:EXECute**, page 65).

Mini-Program Stops – Wait Encountered

Whenever the mini-program encounters a wait, the type of wait is displayed in the prompt line. The four mini-program wait types are given in Table 4-5.

Mini-Program Errors

When an error is encountered during mini-program execution, the program aborts. The GP700 generates the following error messages:

- The message "ABORTED, <*error_type*>, <*line_number*>" flashes on the display.
- The front-panel ERR indicator lights.
- The abort message is placed in the help page. Press the **HELP** key to get additional information and clear the error.
- The abort message is placed in the remote error queue. Use the remote System Error Query command (see **SYSTem:ERRor?**, page 76) to access the remote error queue.

Aborting a Mini-Program

Press the **ABRT** softkey to abort the currently executing mini-program. Although program execution stops immediately, it is possible that commands already in a module buffer will continue to execute. The GP700 generates an error message ("MINI-PROGRAM ERROR, 905") indicating that a user abort was requested. The error message simply means that program execution was halted externally rather than programmatically.

To abort a mini-program via a remote interface, use the Abort Mini-Program remote command (**PROGram:ABORT**, page 64).

Chapter 5
Remote Operation

This chapter contains information about:

- the GP700 remote interfaces
- local, remote, and local-lockout modes
- capturing errors during remote operation
- command timing
- implementing mini-programs remotely

Remote Interfaces

The GP700 supports two remote interfaces: GPIB (parallel) and RS-232 (serial). Some remote commands are available for only one interface. This chapter contains a short introduction to each form of interfacing, followed by interface-specific configuration information. The chapter ends with a discussion of entering and controlling GP700 mini-programs remotely. The GP700 remote command set is defined in Chapter 6, "Remote Commands".

To remotely program the GP700, your system normally has to handle three levels of programming languages. The three languages are:

1. A top-level programming language resident on the controller such as Microsoft Visual Basic, HP BASIC, Borland C, LabVIEW, etc.
2. An interface driver that translates the controller language into a form that can be used over the bus. The driver is normally provided by the manufacturer of the system interface hardware, and typically provides you with a small set of additional commands for your programming language. The driver must agree with your top-level programming language.
3. A device-specific language resident on the device (e.g. the GP700). The device-specific language is also called the device's command set. This command set is generic to all programming languages and to all interface drivers.

For more information about your top-level programming language and device driver, refer to the documentation provided with your interface card. The GP700 command set remains constant regardless of your controller configuration.

GPIB Interface Programming

GPIB is the acronym for General Purpose Interface Bus (frequently referred to as "the bus"). The bus is described in detail in ANSI/IEEE Standards 488.1-1987 and 488.2-1992. These combined standards control the entire interface mechanically, electrically, and functionally, and specifies a set of data codes, formats, and commands common to all devices. Except for some basic background information, intimate knowledge of the bus is not required to make use of the interface.

GPIB Basics

Talkers, Listeners, Controllers, and Devices

The GPIB bus connects talkers, listeners, controllers, and devices. *Talkers* send messages out over the bus and *listeners* receive messages from the bus. A *Controller* manages the flow of data through the bus by talking and listening to devices. *Devices* talk and listen, but typically do not control the bus. Usually a controller is a computer and a device is a piece of test instrumentation.

The GP700 is a device. It talks and listens, but cannot control the bus. In order to remotely interface to the GP700 you must provide some type of controller.

The System Interface

The system interface is the component that connects a controller or device to the bus. The GP700 comes with an IEEE488.2 system interface built-in. To access the bus, connect a GPIB cable from the bus to the GP700 rear-panel GPIB connector (see Figure 2-7). You must provide an interface for your controller to the bus. This is typically called an interface board or card.

Managing Information on the Bus

With a controller, a system interface, and a device in place, all that's left is a method of directing information. In order to manage the flow of messages and responses, two addresses are required:

- The system interface address
- The device address

The system interface address is sometimes called the bus address or the board index. It is typically set by the controller's interface board (by a set of DIP switches, jumpers, etc.). The device address is called the primary address of the device. The primary address is factory-set to 3. The GP700 does not support a secondary address.

GPIB Command Structure

A GP700 program message consists of an interface command, an interface address, a device primary address, a GP700 command, and a message terminator. In addition, several GP700 commands may be combined into one command message through use of a message separator.

The structure of a top-level GPIB command sent using a PC-compatible controller equipped with a National Instruments GPIB interface board running Microsoft Visual Basic is

```
CALL Send(<board>, <address>, "<command>", <terminator>)
```

where

CALL Send() is a Visual Basic command that sends messages over the bus,

board is the board index or interface address,

address is the primary address of the GP700 on the bus,

command is the GP700 command (enclosed in quotation marks),

terminator marks the end of the command.

The **CALL Send()** instruction is defined by the interface driver. The National Instruments interface board comes with drivers for Visual Basic and other top-level languages. There may be some variation in command syntax depending on your particular interface board and drivers. For more information about the requirements of your system, refer to the documentation provided with your interface card.

For example, the Visual Basic command to move module M1 to channel 17, and module S1 to channel 2 is

```
CALL Send(0, 3, "M1 17; S1 2", NLend)
```

where

CALL Send() is a command that sends messages over the bus from within Visual Basic programs,

"0" is the default board index (refer to the documentation provided with your GPIB interface board),

"3" is the default primary address for the GP700,

"M1 17" is the GP700 command to move module M1 to channel 17,

";" is the message separator,

"S1 2" is the second GP700 command,

"NLend" is the mnemonic provided by the interface driver for the newline character (ASCII decimal 10), and serves as the message terminator.

GPIB Addressing

The factory set default GPIB address for the GP700 is 3. The GPIB address may be verified or changed from the front panel (see "Changing the GPIB Address", page 20).

GPIB Terminator

The GP700 message terminator is the newline character (ASCII decimal 10). For the purpose of this manual, we use the National Instruments GPIB mnemonic for a newline character, "NLend", for all GPIB code examples in this manual.

GPIB Queries

When you send a query command (a command which requests information from the GP700) via the GPIB interface, the requested information is placed in the GP700 GPIB output queue. You must then instruct your GPIB interface board to read and clear the output queue. This command is provided by your controller.

The output queue holds only one message. If you send a second query command before reading the results of the first, the output queue will be overwritten and the GP700 will generate an error message.

IEEE 488 Compliance Criteria

In compliance with IEEE 488.1 and 488.2, the GP700 has the capabilities listed in Table 5-1.

Table 5-1: *GP700 IEEE488 Specification Compliance*

IEEE 488.1 Interface Function	IEEE 488.1 Subsets
Source Handshake	SH1
Acceptor Handshake	AH1
Talker	T6
Listener	L4
Service Request	SR1
Remote Local	RL1
Parallel Poll	PP0
Device Clear	DC1
Device Trigger	DT1
Controller	C0
Electrical Interface	E1

RS-232 Interface Programming

Serial programming of the GP700 is accomplished using the RS-232 interface. The RS-232 standard partially dictates the mechanical, electrical and functional specification for the interface, but leaves a lot open for customization by each user. Fortunately the variables that require adjusting are few and easily accessible using industry-standard interfaces and programming languages.

RS-232 Basics

What's On The Interface?

The interface connects a single controller to a single device. The RS-232 controller is called the Data Communication Equipment (DCE). The device is called the Data Terminal Equipment (DTE). The DCE does not necessarily control the interface. Rather, it directs the activity of the DTE. The only exception to this rule is when handshaking is implemented (the GP700 does not implement handshaking). In the case of the GP700, the DCE is usually a computer. The DTE is the GP700.

The System Interface

The system interface is the component that connects the DCE to the DTE. It is usually made up of a controller RS-232 interface and a device RS-232 interface.

The GP700 comes equipped with a built-in RS-232 interface. To access the interface, connect a properly wired cable from your DCE to the rear-panel RS-232 receptacle on the GP700. Your DCE may have a 9-pin or 25-pin RS-232 connector. Figure 5-1 shows the proper mating connections for both cases. You must provide the cable and controller interface (normally a COM port on a PC).

Managing Information on the Interface

Because each link between the DCE and the DTE is unique and exclusive, no device address is required. Instead you simply send commands to the controller's interface. To ensure that commands are sent successfully, you must make sure that the two interfaces are properly configured (see below).

RS-232 Interface Configuration

The GP700's RS-232 interface configuration is given in Table 5-2. The baud rate is adjustable from the front panel (see "Changing the RS-232 Baud Rate", page 20) and comes factory set at 9600.

Table 5-2: *RS-232 Configuration Parameters*

Parameter	Value
baud rate	9600 (default) or 1200
number of bits	8
number of stop bits	1
parity	none

RS-232 Rear Panel Receptacle Configuration

The GP700 is equipped with a built-in RS-232 interface. The GP700 back-panel RS-232 receptacle is shown in Figure 5-1. The RS-232 line assignments are defined in Table 5-3.

Figure 5-1: *RS-232 Receptacle*

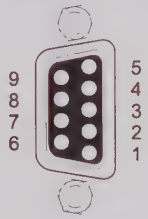
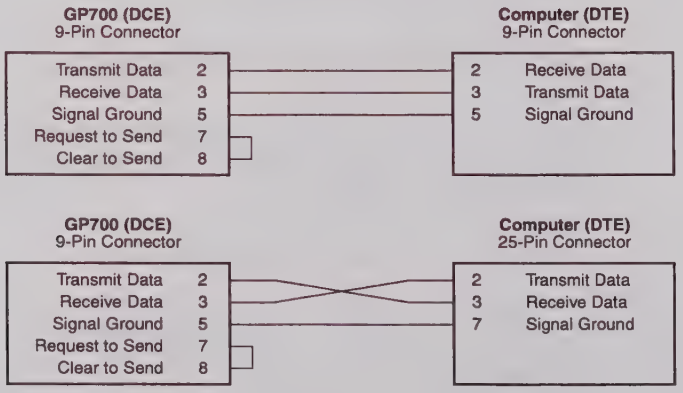


Table 5-3: *RS-232 Line Assignments*

Line Number	Signal Description
1	no connect
2	transmit data
3	receive data
4	no connect
5	signal ground
6	no connect
7	internal jumper to line 8
8	internal jumper to line 7
9	no connect

To access the interface connect a properly wired cable from your controller to the RS-232 rear-panel receptacle. RS-232 connectors and cables typically come in 9-pin or 25-pin configurations. Figure 5-2 shows the proper mating connections for both cases.

Figure 5-2: *RS-232 Cable Configuration*



RS-232 Command Structure

An RS-232 program message typically consists of an interface command, a GP700 command, and a command terminator. Several GP700 commands may be combined into one command message through the use of a message separator. You must send the entire RS-232 command message at once. If there is a pause between characters in a command message, the GP700 may generate a syntax error and start looking for a new command.

The structure of a top-level RS-232 command sent using a PC-compatible controller running Microsoft Visual Basic Professional Edition is

```
Comm1.Output = "<command>" + <terminator>
```

where

Comm1.Output is a command that sends messages to a device from within Visual Basic programs,

command is the GP700 command (enclosed in quotation marks),

terminator marks the end of the command.

The **Comm1.Output** command is provided with the Microsoft Visual Basic Professional Edition drivers, and is enabled when you use the Communications Control. The Communications Control allows you to use a PC COM port as the controller interface.

For example, the Visual Basic command for sending M-type module M1 to channel 17, and S-type module S1 to channel 2 is

```
Comm1.Output = "M1 17; S1 2" + Chr$(10)
```

where

Comm1.Output is a command that sends messages to a device from within Visual Basic programs,

"`M1 17`" is the first GP700 command,

"`;`" is the message separator,

"`S1 2`" is the second GP700 command,

"`Chr$(10)`" is a newline character (ASCII decimal 10), and serves as the message terminator.

RS-232 Terminator

The GP700 RS-232 message termainator is the new-line character (ASCII decimal 10).

RS-232 Queries

When you send a query command (a command which requests information from the GP700) via the RS-232 interface, the GP700 places the queried information into the GP700 RS-232 output buffer, where it automatically shifts out to the controller's input queue at the set baud rate. It may be necessary to add a small delay between sending a query command and reading the controller's input queue to allow the queried information to transmit completely. Because the GP700 does not implement handshaking, data will be lost if the controller is not ready to receive data following a query.

If you send a second query before the GP700 is able to shift out the requested data, the GP700 will overwrite the previous data, corrupting the output buffer. No error message will be generated.

Error Messages

The GP700 responds with the following error conditions when an error occurs:

- The front-panel ERR indicator lights
- An error message is placed in the remote error queue and on the help screen.
- One of four error bits is set in the Standard Event Status Register

To investigate the most recent error, press the **HELP** to see the error message on the front-panel display. Use the System Error Query command (see **SYSTem:ERRor?**, page 76) to read and remove the topmost error from the remote error queue. Failure to read and remove errors from the remote error queue may cause the queue to overflow. In this case the GP700 will log an overflow error and overwrite some error data. The error codes are defined in Table 7-2, "Remote Error Codes", page 83.

It may be necessary to return the GP700 to local mode before you are able to use the **HELP** function from the front panel. You can return the GP700 to local mode remotely using the Close RS-232 Interface command (see **SERIAL:CLOSE**, page 71) or by pressing the **LOCAL** key.

Capturing Error Events

There are several ways to determine if an error has occurred:

- Use the System Error Query command (see **SYSTem:ERRor?**, page 76) to read and remove an error message from the GP700's error queue. The error message consists of an error code and an error description. This method is available via GPIB and RS-232.
- Read the Standard Event Status Register using the Event Status Register Query command (see ***ESR?**, page 53). If an error has occurred, one of the four error bits in the Standard Event Status Register will be set to 1. This method is available via GPIB.
- Use service requests and serial polling to determine if an error has occurred. When properly configured, the GP700 sends a service request signal on the bus when it detects an enabled event such as an execution error. The controller identifies the source of the service request by polling all devices on the bus. For information about serial polling, see the documentation for your GPIB interface controller. To enable one or more of the Standard Event Status Register error bits for service requests, use the Event Status Enable command (see ***ESE**, page 52) and the Service Request Enable command (see ***SRE**, page 71).

Local, Remote, and Local Lockout Modes

There are three levels of front-panel keypad control: local mode, remote mode, and local lockout mode. Remotely accessing the GP700 can affect the level of control available from the front-panel keypad. While in local mode, the GP700 can be controlled from the front panel. In remote mode, the GP700 ignores most keypad entry. You can exit remote mode by pressing the **LOCAL** key. In local lockout mode, the GP700 ignores all keypad entry until it is released remotely.

Upon communicating with the GP700 via the GPIB interface, the GP700 enters remote mode and lights the front-panel RMT indicator. GPIB messages passed to the GP700 will light the front-panel LSN indicator, and messages sent by the GP700 will light the front-panel TLK indicator.

The GP700 also enters remote mode when you send an Open RS-232 Interface command (see **SERIAL:OPEN**, page 71) to the GP700 via the RS-232 interface. There is no indication on the front-panel display that the device has entered remote mode. To unlock the front panel, press **LOCAL** or send the Close RS-232 Interface command remotely (see **SERIAL:CLOSE**, page 71).

The GP700 enters local lockout mode when you send a GPIB **Local Lockout** message. To release the GP700 from local lockout mode, send a GPIB Enable Local message or cycle power to the device. You cannot exit local lockout mode by pressing the **LOCAL** key. For more information about the Local Lockout and Enable Local routines, refer to the documentation provided with your GPIB controller.

Command Timing

This section suggests strategies for avoiding command collision errors when sending multiple commands to the GP700 in rapid succession. Skip this section if you don't expect command collision problems.

The purpose of command synchronization is to avoid sending commands to busy modules. Sending a command to a busy module will result in an error. On the other hand, it is legal to send a command to a non-busy module even when other component modules are busy. Note that because two-position switch modules are usually configured in banks, if a single two-position switch module in a bank is busy then the whole bank is busy. See Appendix B, "Modules" for more information about module banks.

There are several methods for synchronizing commands to prevent busy errors. Command synchronizing strategies include:

- Monitor the Status Byte directly to determine if any modules are busy. Use the Status Byte Query command (see ***STB?**, page 73) to read the GP700 Status Byte. Bit 0 indicates whether any GP700 component modules are busy. There is a short processing latency following the receipt of a command before the Busy bit transitions to busy status. If you query status during this latency period, you will receive a false-ready response. This method is available via GPIB.
- Use the Operation Complete Query command (see ***OPC?**, page 63) to determine if a specific module operation is complete. The command places a "1" in the output queue when the operation completes. This method is available via GPIB and RS-232.

- Monitor the Standard Event Status Register directly to determine if a specific operation is complete. Activate the Operation Complete (OPC) bit of the Standard Event Status Register using the Operation Complete command (see ***OPC**, page 63). Read the ESR using the Event Status Register Query command (see ***ESR?**, page 53). If the OPC bit has been activated, it transitions to 1 when the targeted event is complete. This method is available via GPIB.
- Use service requests and serial polling to determine if a specific module operation is complete. When properly configured, the GP700 sends a service request signal on the bus when it detects an enabled event such as the completion of a module operation. The controller identifies the source of the service request by polling all devices on the bus. For information about serial polling, see the documentation for your GPIB interface controller. For information about enabling GP700 service requests, see the Operation Complete command (see ***OPC**, page 63), the Event Status Enable command (see ***ESE**, page 52), and the Service Request Enable command (see ***SRE**, page 71). This method is available via GPIB.
- Use the Wait-To-Continue command (see ***WAI**, page 79) to force the GP700 to buffer commands until all current operations are complete. This method is available via GPIB and RS-232.

These are the easiest and most reliable methods to avoid module busy errors. However, some of these methods can be overly conservative when dealing with a large number of modules. The Busy bit in the Status Byte is set whenever *any module* is busy. Since you may send commands to a *non-busy* module at any time, waiting for the Status Byte to indicate not busy can add a large error margin. While it is possible to determine if a specific operation is complete, there is no direct method for determining which module is busy at any given time.

If you need to send commands more quickly you can:

- Try sending successive commands to a single module as far apart in the command string as possible. In other words, if you need to move four modules, send the commands in sequential order: M1-M2-M3-M4, and then M1-M2-M3-M4. This is far less prone to generating errors than sending the commands in reverse order: M1-M2-M3-M4, and then M4-M3-M2-M1. The second command sent to M4 will probably generate an error, as M4 will most likely be busy with the first operation.
- Use the Save Device State command (see ***SAV**, page 70) and the Recall Device State command (see ***RCL**, page 66) to save and recall frequent settings. The GP700 will coordinate the set commands such that no error is generated.

Calibrating Setting Time

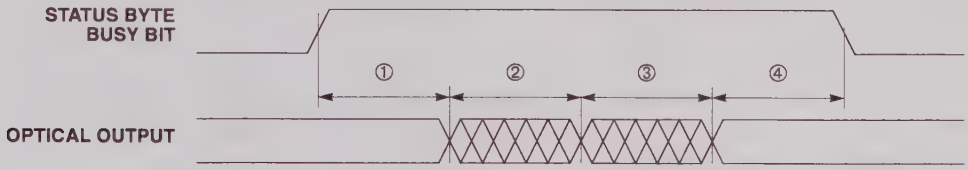
This section suggests strategies for determining precise switching and setting times. Skip this section unless you need this information.

The GP700 busy indicators (the Operation Complete bit of the Standard Event Status Register, the Busy bit of the Status Byte Register, the front-panel BUSY indicator, etc.) indicate when the main processor has determined that all modules have stopped moving and that there are no pending commands. The busy indicators are conservative by design in order to maximize reliability.

The relationship between the active status of the Busy bit of the Status Byte Register and actual setting time is shown in Figure 5-3. The period while the busy bit is high is broken into four constituent periods. The first period represents processing overhead while the GP700 sends the set command to the appropriate module. During this time the optical

connection remains stable. During the second period the module moves from one setting to another. The third period is the debounce period during which the optical connection stabilizes. Finally, the fourth period serves as a buffer to prevent command collision errors.

Figure 5-3: *Set Command Timing*



For users who require more precise knowledge of optical setting start and stop times, the best solution is to calibrate your particular device and interface using a source and detector. The time at which any given module starts motion relative to the time at which the GP700 transitions to busy status is very repeatable (on the order of ±6ms). Furthermore, the time at which the module physically stops moving and the optical power becomes stable relative to the beginning of switch motion is also very repeatable (on the order of ±6ms). Finally, the incremental switching time increase for M- and I-type modules with the number of channels moved is linear (the analogous incremental increase in attenuation or center wavelength for A- and F-type modules, however, is not linear). This means it is possible to calibrate the optical setting time of your particular GP700 to a high degree of accuracy and reliability.

To calibrate switching time, you must take two measurements for each step size of interest. First, measure the period beginning with the GP700 transition to busy status and ending with commencement of switch motion (the end of valid data on the current optical channel).

Second, measure the period beginning with the GP700 transition to busy status and ending with completion of switch motion (the beginning of valid data on the new optical channel). To trap the transition to busy status, use the Service Request Enable command (see ***SRE**, page 71) to enable the Busy bit (bit 0) of the Status Byte to service requests. Note that when switching a multi-channel or matrix switch module to a channel number smaller than the current channel number, the input collimator sweeps past the desired output channel, generating a momentary invalid optical output before the switch operation is complete.

Because any given GPIB controller can show significant variations in service request response time, the performance of your interface can limit the accuracy of your switching time measurements. If you are seeing variations in start and stop times not in line with the repeatability specifications above, your interface is most likely the cause. There are two methods for overcoming this difficulty:

- Break out the SRQ line from your GPIB cable and capture the Busy Bit service request through the same data acquisition system you use to measure the optical sensor output. For example, measure the signals with a multi-channel A/D board with on-board memory. You can dramatically reduce timing variation by using the acquisition board clock rather than your GPIB interface.
- Use a GP700 mini-program to send a GPOUT signal and then execute a step command. Run the GPOUT signal and the sensor signal into the same A/D board mentioned above and measure the same way. Table 5-4 contains a mini-program that can be used to measure switching time.

Table 5-4: Command Synchronization Using General Purpose Outputs

Keystrokes	Prompt Line	Description
PROG **REC**		Enter mini-program record mode.
M **39** **ENTER**	001>>M1:39	Set M1 to channel 39.
GPO **ENTER**	002>>GPOUT	Send a general purpose output.
M **40** **ENTER**	003>>M1:40	Set M1 to channel 40.
GPO **ENTER**	004>>GPOUT	Send a general purpose output.
PROG **SAVE** **0** **ENTER**	>SAVE AS:0	Save mini-program zero and exit record mode.

Implementing Mini-Programs Remotely

To remotely implement a mini-program, use the remote mini-program instruction set (see Table 6-3, "Remote Mini-Program Instruction Set", page 45) in conjunction with the Append New Line to Mini-Program command (see **PROGram:NEWLine**, page 65). The remote mini-program instructions duplicate the functions described in Chapter 4, "Mini-Programs".

The first step in remotely entering a mini-program is to clear a program slot with the Delete Mini-Program command (see **PROGram:DELete**, page 64). Add successive new lines of mini-program instructions using the **PROG:NEWL** command. Each of the commands in the remote mini-program instruction set can be used as the *instruction* parameter to the **PROG:NEWL** command. You must start all mini-programs using the Begin Mini-Program instruction (see **MBEG**, page 62), and end all mini-programs with the End Mini-Program instruction (see **MEND**, page 62).

The following Visual Basic code downloads the mini-program discussed in the manual section titled "Mini-Program Example", page 21.

```
Call Send(0, 3, "PROG1:DEL", NLend)' clear prog 1
Call Send(0, 3, "PROG1:NEWL 'MBEG1'", NLend) ' begin sending
Call Send(0, 3, "PROG1:NEWL '*RCL 1'", NLend) ' recall state 1
Call Send(0, 3, "PROG1:NEWL 'BLOOP 16'", NLend)' loop 16 times
Call Send(0, 3, "PROG1:NEWL 'WAIT K'", NLend) ' keypad wait
Call Send(0, 3, "PROG1:NEWL 'S1 2'", NLend) ' move S1 to 2
Call Send(0, 3, "PROG1:NEWL 'WAIT K'", NLend) ' keypad wait
Call Send(0, 3, "PROG1:NEWL 'INCM1'", NLend) ' increment M1
Call Send(0, 3, "PROG1:NEWL 'INCM2'", NLend) ' increment M2
Call Send(0, 3, "PROG1:NEWL 'S1 1'", NLend) ' move S1 to 1
Call Send(0, 3, "PROG1:NEWL 'ELOOP'", NLend) ' end loop
Call Send(0, 3, "PROG1:NEWL '*RCL 1'", NLend) ' recall state 1
Call Send(0, 3, "PROG1:NEWL 'MEND'", NLend) ' end program 1
```

For the purpose of this manual, single quotes are used to delimit the mini-program *instruction* parameter. Your programming environment may require a different quote delimiter. Both single- and double-quote delimiters conform to the IEEE 488.2 standard.

Chapter 6
Remote Commands

This chapter contains:

- a summary of remote commands supported by the GP700
- a list of configuration assumptions that apply to all code examples in this manual
- definitions and examples for all GP700 commands

Command Set Summary

The GP700 supports three categories of remote commands: IEEE488 common commands, GP700 device-specific commands, and mini-program remote instructions. The following sections list the three command sets.

IEEE488 Common Commands

The IEEE488 common commands are supported in compliance with the IEEE 488.2 specification. The commands are available via the GPIB interface, and in many cases, via the RS-232 interface. These commands are always prefaced by an asterisk (" * "), and are sometimes called "starred commands".

Table 6-1: *IEEE488 Common Command Set*

Command	Page	Description	GPIB	RS-232
***CLS**	49	Clear Status	•	
***ESE**	52	Event Status Enable	•	
***ESE?**	53	Event Status Enable Query	•	
***ESR?**	53	Event Status Register Query	•	
***IDN?**	59	Identification Query	•	•
***OPC**	63	Operation Complete	•	
***OPC?**	63	Operation Complete Query	•	•
***RCL**	66	Recall Device State	•	•
***RST**	67	Reset	•	•
***SAV**	70	Save Device State	•	•
***SRE**	71	Service Request Enable	•	

Table 6-1: IEEE488 Common Command Set (Continued)

Command	Page	Description	GPIB	RS-232
***SRE?**	72	Service Request Enable Query	•	
***STB?**	73	Status Byte Query	•	
***TRG**	78	Trigger	•	•
***TST?**	79	Self-Test Query	•	•
***WAI**	79	Wait-To-Continue	•	•

GP700 Device-Specific Commands

Use GP700 device-specific commands to set and query GP700 component modules and system configurations. Most device-specific commands are available via both the GPIB and RS-232 interfaces. You may preface any device-specific command with a colon (e.g. "**:DECM**").

Table 6-2: GP700 Device-Specific Command Set

Command	Page	Description	GPIB	RS-232
A	48	Attenuation Level Select	•	•
A?	48	Attenuation Level Query	•	•
DECM	50	Decrement M-Type Module	•	•
DISPlay:OFF	50	Turn Off Display	•	•
DISPlay:ON	51	Turn On Display	•	•
EXTernal:CONFig	54	External Trigger Configuration	•	•
EXTernal:CONFig?	55	External Trigger Configuration Query	•	•
F	55	Filter Center Wavelength Select	•	•
F?	56	Filter Center Wavelength Query	•	•
GPOut:CONFig	56	General Purpose Output Configuration	•	•
GPOut:CONFig?	57	General Purpose Output Configuration Query	•	•
GPOUT	58	General Purpose Output	•	•
I	58	Matrix Channel Select	•	•
I?	58	Matrix Channel Query	•	•
INCM	59	Increment M-type Module	•	•
M	60	M-Type Module Channel Select	•	•
M0	60	M-Type Module All Call	•	•
M?	61	M-Type Module Channel Query	•	•
PROGram:ABORT	64	Abort Mini-Program	•	•
PROGram:DELete	64	Delete Mini-Program	•	•
PROGram:EXECute	65	Execute Mini-Program	•	•
PROGram:NEWLine	65	Append New Line To Mini-Program	•	•
PROGram?	66	Program Contents Query	•	•
S	68	S-Type Module Channel Select	•	•
S0	68	S-Type Module All Call	•	•
S?	70	S-Type Module Channel Query	•	•
SERIAL:CLOSE	71	Close RS-232 Interface		•
SERIAL:OPEN	71	Open RS-232 Interface		•
SYSTem:CONFig?	74	System Configuration Query	•	•
SYSTem:DATE	75	Set System Date	•	•

Table 6-2: GP700 Device-Specific Command Set (Continued)

Command	Page	Description	GPIB	RS-232
SYSTem:DATE?	75	System Date Query	•	•
SYSTem:ERRor?	76	System Error Query	•	•
SYSTem:TIME	77	Set System Time	•	•
SYSTem:TIME?	77	System Time Query	•	•
TOGS	78	S-Type Module Toggle	•	•

Remote Mini-Program Instructions

Use the remote mini-program instruction set in conjunction with the Append New Line to Mini-Program command (see **PROGram:NEWLine**, page 65). These instructions exactly duplicate the functions described in Chapter 4, "Mini-Programs". Remote mini-program instructions can be sent via both the GPIB interface and the RS-232 interface.

Table 6-3: Remote Mini-Program Instruction Set

Instruction	Page	Description
A	48	Attenuation level Select
BLOOP	49	Begin Loop
DECM	50	Decrement M-Type Module
ELOOP	51	End Loop
F	55	Filter Center Wavelength Select
GPOUT	58	General Purpose Output
I	58	Matrix Channel Select
INCM	59	Increment M-Type Module
M	60	M-type Module Channel Select
M0	60	M-type Module All Call
MBEG	62	Begin Mini-Program
MEND	62	End Mini-Program
*RCL	66	Recall Device State
S	68	S-Type Module Channel Select
S0	68	S-Type Module All Call
TOGS	78	S-Type Module Toggle
WAIT	80	Wait For Trigger Input

Configuration Assumptions

GPIB Control

All GPIB command examples in this chapter assume that the GPIB controller is a PC-compatible system equipped with a National Instruments GPIB interface board (PCII) running Microsoft Visual Basic Professional Edition, where:

- The GP700 GPIB address is set to the default address of 3.
- The board index of the GPIB interface board is set to the default index of 0.
- The GPIB Visual Basic drivers provided by National Instruments are loaded in the controller.
- A variable called `buffer$` has been dimensioned in Visual Basic. In some instances you may have to dimension the string longer than 256 characters.

```
Dim buffer As String * 256          ' 256 character wide string
```

- A variable `addrlist%` has been dimensioned and initialized in Visual Basic. The variable is used for executing the **EnableLocal** GPIB interface command.

```
Dim addrlist(2) As Integer          ' Create small address
                                    ' list array
addrlist%(0) = 3                    ' Initialize address list with
addrlist%(1) = NOADDR               ' GP700 address only
```

- White spaces shown in the commands are significant.

RS-232 Control

The majority of examples in this manual demonstrate commands using the GPIB interface. Refer to "RS-232 Command Structure", page 37, for information about the structure of equivalent RS-232 commands. One important distinction between RS-232 control and GPIB control is the behavior of query commands. Whereas GPIB query responses are placed into the GPIB output queue, RS-232 query responses are sent automatically to the controller.

All RS-232 command examples in this chapter assume that the RS-232 controller is a PC-compatible system running Microsoft Visual Basic Professional Edition, where:

- The controller is configured per Table 5-2. The following code configures the controller programmatically:

```
Comm1.CommPort = 2                  ' Use COM port 2 for RS232
Comm1.Settings = "9600,N,8,1"       ' 9600 baud, no parity
                                    ' 8 bits, 1 stop bit
Comm1.PortOpen = True               ' Open the port
```

- A variable called `buffer$` has been dimensioned in Visual Basic. In some instances you may have to dimension the string longer than 256 characters.

```
Dim buffer As String * 256          ' 256 character wide string
```

- White spaces shown in the commands are significant.

Command Conventions

Names

Some command names have both short and long forms. For each command name, the parts given in upper-case characters are required. The parts given in lower-case characters are optional. All commands are case insensitive. For example, all of the following commands are valid:

```
SYST:ERR?
SYSTEM:ERROR?
SYST:ERROR?
SYSTEM:ERR?
sYsTem:ErrOR?
```

Availability

Every command definition has an application summary box indicating whether a command is available via the GPIB interface, via the RS-232 interface, or in the context of remote mini-programs. For example, the following application summary box indicates that the command is available via GPIB and RS-232, but cannot serve as a remote mini-program instruction.

Figure 6-1: *Sample Application Summary Box*

Some commands function only in the context of mini-programs. Although such commands are downloaded via the GPIB or RS-232 interfaces using the Add New Line to Mini-Program command (see **PROGram:NEWLine**, page 65), they cannot serve as top-level GPIB or RS-232 commands.

Parameters

Some GP700 commands have required and/or optional parameters. All parameter names are in *italics*. In command strings, parameter names are enclosed in angle brackets ("<>"). Optional parameters are enclosed in braces ("{}").

Command Definitions

A
Attenuation Level Select

Application: GPIB | RS-232 | MINI

Format: A<*module*> <*attenuation*>

Description: The **A** command sets the level of attenuator module number *module* to the value of *attenuation*. Depending on the configuration of your device, the *attenuation* parameter can be any floating point value in the range 0.00 – 80.00dB. Attenuation values are rounded to the nearest 0.01dB. The actual maximum attenuation may vary from device to device. Use the System Configuration Query command (**SYSTem:CONFig?**) to obtain the actual attenuation range(s) of your module(s). Do not send set commands to busy modules.

Example: The following Visual Basic command sets attenuator module A2 to an attenuation level of 5.14dB.

```
CALL Send(0, 3, "A2 5.14", NLend)
```

See Also: **SYSTem:CONFig?**, page 74
"Command Timing", page 39

A?
Attenuation Level Query

Application: GPIB | RS-232 | MINI

Format: A<*module*>?

Description: The **A?** command places the current setting of attenuator module number *module* into the output queue. The returned two-place-decimal string is the attenuation in dB.

Example: The following Visual Basic code places the current setting of attenuator module A2 into the output queue, and reads the output queue into `buffer$`.

```
CALL Send(0, 3, "A2?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "`34.53`" would indicate that attenuator module A2 is set to 34.53dB.

BLOOP
Begin Loop

Application: GPIB | RS-232 | **MINI**

Format: `BLOOP <count>`

Description: The **BLOOP** instruction begins a looping sequence in a mini-program. Mini-program loops begin with the **BLOOP** instruction and end with the End Loop instruction (**ELOOP**). When the loop begins, the loop counter is initialized with the value *count*. Upon reaching the **ELOOP** instruction, the loop counter is decremented and compared to zero. The loop repeats if the loop counter is not equal to zero. Otherwise, the loop ends and the mini-program continues with the next instruction.

To create an infinite loop, set the *count* to zero. An infinite loop will repeat until you press the **ABRT** softkey, or until you send an Abort Mini-Program command (**PROGram:ABORT**). You may nest loops four deep. You must have an **ELOOP** instruction for every **BLOOP** instruction. An unmatched loop instruction will cause a run-time error.

Example: The following Visual Basic command appends a Begin Loop instruction to mini-program 1. The instruction begins a loop with a loop count of 16.

```
CALL Send(0, 3, "PROG1:NEWL 'BLOOP 16'", NLend)
```

See Also: **ELOOP**, page 51
PROGram:ABORT, page 64
PROGram:NEWLine, page 65
"Setting Up a Program Loop", page 27

**CLS*
Clear Status

Application: **GPIB** | RS-232 | MINI

Format: `*CLS`

Description: The ***CLS** command clears the Standard Event Status Register, the Status Byte Register, and the error queue.

Additionally, if sent immediately following a message terminator (on its own line, not part of a multiple command), the ***CLS** command also clears the output queue and the Message Available (MAV) bit of the Status Byte Register.

Example: The following Visual Basic command sends a Clear Status instruction.

```
CALL Send(0, 3, "*CLS", NLend)
```

See Also: Table 6-5, "Standard Event Status Register", on page 54
Table 6-10, "Status Byte Register", on page 73

DECM

Decrement M-Type Module Channel

Application: GPIB | RS-232 | MINI

Format: DECM<*module*>{ <*port*>}

Description: The **DECM** command decrements multi-channel switch module number *module* by one channel. The module port is indicated by *port*. Output channels pass through *port* "A" and input channels pass through *port* "B". To decrement the input channel of an M-type module with multiple input settings, specify *port* "B". If no port is specified, *port* "A" is decremented. Do not send switching commands to busy modules.

Example: The following Visual Basic code decrements the input channel of multi-channel switch module M2.

```
CALL Send(0, 3, "DECM2 B", NLend)
```

See Also: "Command Timing", page 39

DISPlay:OFF

Turn Off Display

Application: GPIB | RS-232 | MINI

Format: DISP:OFF

Description: The **DISP:OFF** command inactivates display updates, speeding command processing and execution. You can reactivate the display by sending the Turn On Display command (**DISP:ON**) or pressing **LOCAL** on the front-panel keypad.

Example: The following Visual Basic command turns off the front-panel display.

```
CALL Send(0, 3, "DISP:OFF", NLend)
```

DISPlay:ON
Turn On Display

Application: **GPIB** **RS-232** MINI

Format: `DISP:ON`

Description: The **DISP:ON** command activates the display after it has been inactivated using **DISP:OFF**.

Example: The following Visual Basic command turns on the front-panel display.

```
CALL Send(0, 3, "DISP:ON", NLend)
```

ELOOP
End Loop

Application: GPIB RS-232 **MINI**

Format: `ELOOP`

Description: The **ELOOP** instruction ends a looping sequence in a mini-program. Mini-program loops begin with the Begin Loop Instruction (**BLOOP**) and end with the **ELOOP** instruction. Upon reaching the **ELOOP** instruction, the loop counter is decremented and compared to zero. The loop repeats if the loop counter is not equal to zero. Otherwise, the loop ends and the mini-program continues with the next instruction. You must have an **ELOOP** command for every **BLOOP** command. An unmatched loop instruction will cause a run-time error.

Example: The following Visual Basic command appends an End Loop instruction to mini-program 1.

```
CALL Send(0, 3, "PROG1:NEWL 'ELOOP'", NLend)
```

See Also: **BLOOP**, page 49
PROGram:NEWLine, page 65
"Setting Up a Program Loop", page 27

*ESE

Event Status Enable

Application: GPIB RS-232 MIN

Format: `*ESE <mask>`

Description: The ***ESE** command assigns the value of *mask* to the Standard Event Status Enable Register. Every bit in the Standard Event Status Enable Register corresponds to a specific event. The events are defined in Table 6-4. When a enabled event takes place, the Event Summary Bit (ESB) of the Status Byte Register is set to 1. To enable an event bit, set the bit to 1. Set disabled bits to 0.

To determine if an enabled event has occurred, execute a Status Byte Query command (***STB?**) and evaluate the ESB. To determine which event took place, execute an Event Status Register Query command (***ESR?**). To read the current mask, execute an Event Status Enable Query command (***ESE?**).

Table 6-4: *Standard Event Status Enable Register*

Name	Description	Bit	Decimal
OPC	Operation Complete	0	1
RQC	Request Control (N/A, always 0)	1	2
QYE	Query Error	2	4
DDE	Device Dependent Error	3	8
EXE	Execution Error	4	16
CME	Command Error	5	32
URQ	User Request	6	64
PON	Power On	7	128

Example: The following Visual Basic command enables a mask to trap the Operation Complete (OPC) and Device Dependent Error (DDE) events (1 + 8 = 9).

```
CALL Send(0, 3, "*ESE 9", NLend)
```

See Also: ***STB?**, page 73
***ESR?**, page 53
***ESE?**, page 53

**ESE?*

Event Status Enable Query

Application: GPIB RS-232 MINI

Format: `*ESE?`

Description: The ***ESE?** command places the contents of the Standard Event Status Enable Register into the output queue. Every bit in the Standard Event Status Enable Register corresponds to a specific event. The events are defined in Table 6-4. Enabled event bits are set to 1. Masked bits are set to 0.

Example: The following Visual Basic code places the contents of the Standard Event Status Enable Register into the output queue and then reads the output queue into `buffer$`.

```
CALL Send(0, 3, "*ESE?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "20" would indicate that the Query Error (QYE) and Execution Error (EXE) bits are enabled (4 + 16 = 20).

See Also: Table 6-4, "Standard Event Status Enable Register", page 52

**ESR?*

Event Status Register Query

Application: GPIB RS-232 MINI

Format: `*ESR?`

Description: The ***ESR?** command reads and clears the contents of the Standard Event Status Register. The result is placed in the output queue. Every bit in the Standard Event Status Register corresponds to a specific event. The events are defined in Table 6-5.

When an event takes place, the appropriate Standard Event Status Register bit is set. If you have enabled the corresponding bit in the Standard Event Status Enable Register using the Event Status Enable command (***ESE**), the Event Summary Bit (ESB) of the Status Byte Register is also set to 1.

One important use of this command is to determine if the GP700 is busy. If the Standard Event Status Register Operation Complete (OPC) bit has been activated with the Operation Complete command (***OPC**), the OPC bit transitions from 0 to 1 upon completion of an operation.

Table 6-5: Standard Event Status Register

Name	Event	Bit	Decimal
OPC	0 - Enabled operation not complete. 1 - Enabled operation completed.	0	1
RQC	0 - Always set to zero (GP700 cannot be a controller)	1	2
QYE	0 - No error reading the output queue. 1 - Output queue read error.	2	4
DDE	0 - No device dependent error. 1 - Hardware failure.	3	8
EXE	0 - No execution error. 1 - Out-of-range module setting.	4	16
CME	0 - No command error. 1 - Message received with bad syntax.	5	32
URQ	0 - No keyboard entry detected. 1 - Keyboard entry detected.	6	64
PON	0 - No power on signal detected. 1 - Power on signal detected.	7	128

Example: The following Visual Basic code places the contents of the event status register into the output queue and then reads the output queue into `buffer$`.

```
CALL Send(0, 3, "*ESR?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "`20`" would indicate that a Query Error (QYE) and an Execution Error (EXE) had taken place since the previous ***ESR?** command.

See Also: ***ESE**, page 52
***OPC**, page 63

EXTernal:CONFig
External Trigger Configuration

Application: GPIB RS-232 MINI

Format: `EXT:CONF <type>, <polarity>`

Description: The **EXT:CONF** command configures the GP700 external trigger port. The trigger *type* can be set to either pulse or level ("`PUL`" or "`LEV`"). The *polarity* of pulse triggers can be set to rising edge or falling edge ("`RISE`" or "`FALL`"). The *polarity* of level triggers can be set to high or low ("`POS`" or "`NEG`"). The triggers are relevant only in the context of mini-program waits.

Example: The following Visual Basic command configures the external trigger port to recognize rising-edge pulse triggers.

```
CALL Send(0, 3, "EXT:CONF PUL, RISE", NLend)
```

See Also: "Configuring the Input/Output Ports", page 19
"Waiting for a Trigger", page 27

EXTernal:CONFig?
External Trigger Configuration Query

Application: GPIB RS-232 MINI

Format: `EXT:CONF?`

Description: The **EXT:CONF?** command places the current configuration of the GP700 external trigger port into the output queue. The command returns

<type>, <polarity>

where *type* is the trigger type and *polarity* is the trigger polarity. A trigger *type* of "`PUL`" indicates a pulse trigger. A trigger *type* of "`LEV`" indicates a level trigger. Pulse triggers have either rising-edge or falling-edge *polarity* ("`RISE`" or "`FALL`"). Level triggers have either high or low *polarity* ("`POS`" or "`NEG`").

Example: The following Visual Basic code places the current external trigger configuration into the output queue, and reads the output queue into `buffer$`.

```
CALL Send(0, 3, "EXT:CONF?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "`LEV, NEG`" would indicate that the external trigger port is configured to recognize low-polarity level-type triggers.

See Also: "Configuring the Input/Output Ports", page 19

F
Filter Center Wavelength Select

Application: GPIB RS-232 MINI

Format: `F`*<module> <wavelength>*

Description: The **F** command sets the center wavelength of filter module number *module* to *wavelength*. The *wavelength* parameter can be any floating point value. Wavelengths are rounded to the nearest 0.01 nm. Wavelength ranges are calibrated individually for each unit and may vary from module to module. Use the System Configuration Query command (**SYSTem:CONFig?**) to determine the range of your module(s). Do not send set commands to busy modules.

Example: The following Visual Basic command sets the center-wavelength of filter module F2 to1546.34nm.

```
CALL Send(0, 3, "F2 1546.338", NLend)
```

See Also: **SYSTem:CONFig?**, page 74
"Command Timing", page 39

F?
Filter Center Wavelength Query

Application: GPIB RS-232 MINI

Format: F<*module*>?

Description: The **F?** command places the current center-wavelength setting of filter module number *module* into the output queue. The returned two-place-decimal string is the center wavelength in nm.

Example: The following Visual Basic code places the current center-wavelength setting of filter module F1 into the output queue, and then reads the output queue into `buffer$`.

```
CALL Send(0, 3, "F1?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "1546.34" would indicate that filter module F1 is set to 1546.34nm.

GPOut:CONFig
General Purpose Output Configuration

Application: GPIB RS-232 MINI

Format: GPO:CONF <*type*>, <*polarity*>

Description: The **GPO:CONF** command configures the GP700 general purpose output port. The output signal *type* can be set to either pulse or level ("PUL" or "LEV"). The *polarity* of pulse signals can be set to rising edge or falling edge ("RISE" or "FALL"). The *polarity* of level signals can be set to high or low ("POS" or "NEG"). General purpose output signals can be generated remotely and within mini-programs using the General Purpose Output command (**GPOUT**).

Example: The following Visual Basic command configures the general purpose output port to send rising-edge pulse signals.

```
CALL Send(0, 3, "GPO:CONF PUL, RISE", NLend)
```

See Also: **GPOUT**, page 58
"Configuring the Input/Output Ports", page 19
"Sending a General Purpose Output", page 28

GPOut:CONFig?

General Purpose Output Configuration Query

Application: GPIB RS-232 MINI

Format: `GPO:CONF?`

Description: The **GPO:CONF?** command places the current configuration of the GP700 general purpose output port into the output queue. The command returns

<type>, <polarity>

where *type* is the output signal type and *polarity* is the signal polarity. A signal *type* of "`PUL`" indicates a pulse signal. A signal *type* of "`LEV`" indicates a level signal. Pulse signals have either rising-edge or falling-edge *polarity* ("`RISE`" or "`FALL`"). Level signals have either high or low *polarity* ("`POS`" or "`NEG`").

Example: The following Visual Basic code places the current general purpose output trigger configuration into the output queue, and reads the output queue into `buffer$`.

```
CALL Send(0, 3, "GPO:CONF?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "`PUL, RISE`" would indicate that the general purpose output port is configured to generate a rising-edge pulse-type signals.

See Also: **GPOUT**, page 58
"Configuring the Input/Output Ports", page 19
"Sending a General Purpose Output", page 28

GPOUT
General Purpose Output

Application: GPIB | RS-232 | MINI

Format: `GPOUT`

Description: The **GPOUT** command sends an output trigger via the rear-panel general purpose output port according to the configuration set by the General Purpose Output Configuration command (**GPOut:CONFig**) or the front-panel setting.

Example: The following Visual Basic command sends an output trigger over the general purpose output port.

```
CALL Send(0, 3, "GPOUT", NLend)
```

See Also: "Configuring the Input/Output Ports", page 3-19
GPOut:CONFig, page 56
"Sending a General Purpose Output", page 28

I
Matrix Channel Select

Application: GPIB | RS-232 | MINI

Format: `I<input> <output>`

Description: The **I** command establishes an optical connection between matrix input channel *input* and output channel *output*. Do not send switching commands to busy matrix inputs or outputs.

Example: The following Visual Basic command establishes an optical connection between input channel 5 and output channel 8.

```
CALL Send(0, 3, "I5 8", NLend)
```

See Also: "Command Timing", page 39

I?
Matrix Channel Query

Application: GPIB | RS-232 | MINI

Format: `I<input>?`

Description: The **I?** command queries a matrix module and places the channel number of the output connected to input channel *input* into the GP700 output queue.

Example: The following Visual Basic code places the current setting of matrix input 3 into the output queue, and reads the output queue into `buffer$`.

```
CALL Send(0, 3, "I3?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "15" would indicate that matrix input 3 is connected to output 15.

*IDN?
Identification Query

Application: GPIB RS-232 MINI

Format: `*IDN?`

Description: The ***IDN?** command places the GP700's unique identification string into the output queue. The command returns

`DiCon Fiberoptics Inc, GP700, 0, Version <revision>`

where *revision* is the firmware revision level.

Example: The following Visual Basic code places the GP700 identification string into the output queue, and reads the output queue into `buffer$`.

```
CALL Send(0, 3, "*IDN?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

INCM
Increment M-Type Module

Application: GPIB RS-232 MINI

Format: `INCM<module>{ <port>}`

Description: The **INCM** command increments multi-channel switch module number *module* by one channel. The module port to be incremented is indicated by *port*. Output channels pass through *port* "A" and input channels pass through *port* "B". To increment the input channel of a multi-channel switch module with more than one input setting, specify *port* "B". If no port is specified, *port* "A" is incremented. Do not send switching commands to busy modules.

Example: The following Visual Basic command increments the output channel of multi-channel switch module M2 by one channel.

```
CALL Send(0, 3, "INCM2", NLend)
```

See Also: "Command Timing", page 39

M
Select M-Type Module Channel

Application: GPIB | RS-232 | MINI

Format: M<*module*> <*output*>{, <*input*>}

Description: The **M** command sets multi-channel switch module number *module* to output channel *output* and input channel *input*. The *output* parameter is required for all M-type module configurations. The *input* parameter is optional for M-type modules with more than one input setting. Channel 1 is the default *input* channel if none is specified. Do not send switching commands to busy modules.

Example: The following Visual Basic code sets the output channel of multi-channel switch module M1 to channel 82. If M1 has multiple input settings, the input channel will default to channel 1.

```
CALL Send(0, 3, "M1 82", NLend)
```

The following Visual Basic code sets multi-channel switch module M1 to output channel 16 and input channel 3.

```
CALL Send(0, 3, "M1 16, 3", NLend)
```

See Also: "Command Timing", page 39

M0
M-Type Module All Call

Application: GPIB | RS-232 | MINI

Format: M0 <*output*>{, <*input*>}

Description: The **M0** command synchronously sets all multi-channel switch modules to output channel *output* and input channel *input*. The command moves all M-type modules to the same channel, at the same time, allowing you to use a GP700 as a multiplexed switch. The **M0** command is available only when all the M-type modules within the GP700 are of the same size and configuration.

The *output* parameter is required for all M-type module configurations. The *input* parameter is optional for M-type modules with more than one input setting. Channel 1 is the default *input* channel if none is specified.

The GP700 synchronizes switch operation at the level of the main on-board processor, simultaneously sending switch commands to all M-type module subprocessors. After the GP700 sends switch instructions to the component modules, switch motion is governed by each individual module subprocessor. Although the subprocessors in like modules are identical, some variation in switching speed does occur. This variation is on the order of microseconds.

The **M0** command is available only when controlling or downloading programs remotely. The command is not available from the front panel. Do not send switching commands to busy modules.

Example: The following Visual Basic command selects output channel 14 for all multi-channel switch modules.

```
CALL Send(0, 3, "M0 14", NLend)
```

See Also: "Command Timing", page 39

M?
M-type Module Channel Query

Application: GPIB | RS-232 | MINI

Format: M<*module*>?

Description: The **M?** command places the current channel settings for M-type module number *module* in the output queue. The command returns

<*output*>,<*input*>

where *output* is the output channel number (port A) and *input* is the input channel number (port B). For M-type modules with only one input setting, the returned *input* value is always "1".

Example: The following Visual Basic code places the channel setting of M-type module M1 into the output queue, and then reads the output queue into buffer$.

```
CALL Send(0, 3, "M1?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a buffer$ value of "12,1" would indicate that M-type module M1 is set to input channel 1 and output channel 12.

MBEG
Mini-Program Begin

Application: GPIB RS-232 **MINI**

Format: MBEG<*program*>

Description: The **MBEG** instruction is the first line of mini-program number *program*. All downloaded mini-programs must begin with the **MBEG** instruction. Before recording a new mini-program, you must delete the currently stored program with the Delete Mini-Program command (**PROG:DEL**).

Example: The following Visual Basic command programs the first line of mini-program 1.

```
CALL Send(0, 3, "PROG1:NEWL 'MBEG1'", NLend)
```

See Also: **PROGram:DELete**, page 64
PROGram:NEWLine, page 65

MEND
Mini-Program End

Application: GPIB RS-232 **MINI**

Format: MEND

Description: The **MEND** instruction marks the end of every mini-program. When the currently executing mini-program reaches an **MEND** instruction, the GP700 sets the Mini-Program End Encountered bit (DIO2) of the Status Byte Register to 1.

Running a mini-program that is not properly terminated with an **MEND** instruction will result in a run-time error.

Example: The following Visual Basic command ends mini-program 1.

```
CALL Send(0, 3, "PROG1:NEWL 'MEND'", NLend)
```

See Also: **PROGram:NEWLine**, page 65

*OPC
Operation Complete

Application: GPIB RS-232 MINI

Format: `*OPC`

Description: The ***OPC** command activates the Operation Complete (OPC) bit of the Standard Event Status Register. If the OPC bit is activated, the GP700 sets the OPC bit to 1 when all pending device operations are complete. The ***OPC** command is a one-time operation. When a completed operation sets the OPC bit, the function deactivates. You can monitor the OPC bit by reading the Standard Event Status Register directly or by enabling the bit to service requests using the Service Request Enable command (***SRE**).

Note that the Operation Complete Query command (***OPC?**) is fundamentally different from the Operation Complete command (***OPC**). While the ***OPC** command sets a status register bit when the operation completes, the ***OPC?** command sends a response directly to the output queue when an operation completes.

Example: The following Visual Basic command moves module M1 to channel 25, M5 to channel 17, M2 to channel 2, and requests that the GP700 set the Standard Event Status Register OPC bit to 1 when the three **M** operations are complete.

```
CALL Send(0, 3, "M1 25; M5 17; M2 2; *OPC", NLend)
```

See Also: ***OPC?**, page 63
***SRE**, page 71
"Command Timing", page 39
Table 6-5, "Standard Event Status Register", page 54

*OPC?
Operation Complete Query

Application: GPIB RS-232 MINI

Format: `*OPC?`

Description: The ***OPC?** command places a "1" into the output queue when all pending device operations are complete.

Attempts to read the command response from the GPIB output queue will hang the bus until the response is placed in the queue. Using RS-232, the result of attempting to read this information prior to a response being placed in the output queue is system dependent.

Note that the Operation Complete Query command (***OPC?**) is fundamentally different from the Operation Complete command (***OPC**). While the ***OPC** command sets a status register bit when the operation completes, the ***OPC?** command sends a response directly to the output queue when an operation completes.

Example: The following Visual Basic code moves M1 to channel 25, places an ASCII "1" in the output queue when the operation is complete, and then reads the output queue into `buffer$`.

```
CALL Send(0, 3, "M1 25; *OPC?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

See Also: ***OPC**, page 63
"Command Timing", page 39

PROGram:ABORT
Abort Mini-Program

Application: GPIB RS-232 MINI

Format: `PROG:ABORT`

Description: The **PROG:ABORT** command aborts the currently running mini-program.

Example: The following Visual Basic command aborts the currently running mini-program.

```
CALL Send(0, 3, "PROG:ABORT", NLend)
```

PROGram:DELete
Delete Mini-Program

Application: GPIB RS-232 MINI

Format: `PROG<program>:DEL`

Description: The **PROG:DEL** command deletes mini-program number *program* from the memory of the GP700. You must delete the currently stored mini-program before remotely recording a new program.

Example: The following Visual Basic command deletes mini-program number 3.

```
CALL Send(0, 3, "PROG3:DEL", NLend)
```

PROGram:EXECute
Execute Mini-Program

Application: GPIB | RS-232 | MINI

Format: PROG<*program*>:EXEC

Description: The **PROG:EXEC** command executes mini-program number *program*. Because mini-programs may require keypad input, DiCon recommends that you send a GPIB **EnableLocal** message to release the GP700 from remote mode after sending the Execute Mini-Program command. If you do not send a GPIB **EnableLocal** message, you may be forced to press **LOCAL** to enable keypad entry.

Example: The following Visual Basic code executes stored mini-program number 3, and then enables keypad entry by releasing the GP700 from remote mode.

```
CALL Send(0, 3, "PROG3:EXEC", NLend)
CALL EnableLocal(0, 3, addrlist())
```

PROGram:NEWLine
Append New Line to Mini-Program

Application: GPIB | RS-232 | MINI

Format: PROG<*program*>:NEWL '<*instruction*>'

Description: The **PROG:NEWL** command appends *instruction* to mini-program number *program*. Use this command to download and record mini-programs remotely. Before recording a new mini-program, you must delete the currently stored program with the Delete Mini-Program command (**PROG:DEL**).

Example: The following Visual Basic command appends a Recall Switch State instruction to mini-program number 1.

```
CALL Send(0, 3, "PROG1:NEWL '*RCL 1'", NLend)
```

See Also: **PROGram:DELete**, page 64
***RCL**, page 66
Table 6-3, "Remote Mini-Program Instruction Set", page 45

PROGram?
Program Contents Query

Application: GPIB RS-232 MINI

Format: PROG<*program*>?

Description: The **PROG?** command places the contents of mini-program number *program* into the output queue. The mini-program is returned as one continuous string with instructions separated by semicolons (";"). If no program is stored in the queried location, the command returns ""No program"".

Example: The following Visual Basic code places the contents of mini-program 0 into the output queue, and reads the output queue into buffer$.

```
CALL Send(0, 3, "PROG0?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a buffer$ value of ""MBEG0;M1 5;S1 2;MEND"" would indicate that program number zero, upon execution, sets multi-channel switch module M1 to output channel 5, and then sets two-position switch module S1 to channel 2.

**RCL*
Recall Device State

Application: GPIB RS-232 MINI

Format: *RCL <*register*>

Description: The ***RCL** command restores the GP700 to the state stored in memory register number *register*. User-configurable memory registers 1–9 can store the channel, center-wavelength, and attenuation settings of all component modules. It is possible to remotely recall a complete or partial device state saved via the front panel.

Recalling memory register zero executes a soft reset of the GP700, placing all modules in the states given in Table 6-6.

Table 6-6: Soft Reset State

Module	Reset State
attenuator modules	minimum loss position
filter modules	maximum CWL position[a]
matrix switch module inputs	channel 0 (optical off)
two-position switch modules	channel 1, off, or bypass[b]
multi-channel switch module outputs (Port A)	channel 0 (optical off)
multi-channel switch module inputs (Port B)	channel 1

a. Filter modules reset to an uncalibrated position slightly beyond the maximum center-wavelength position.
b. The actuation style of 2×2 switch modules may vary from device to device. Refer to your configuration diagram to verify the actuation style of your device.

Example: The following Visual Basic command moves all modules to the positions saved in register 1.

```
CALL Send(0, 3, "*RCL 1", NLend)
```

See Also: ***SAV**, page 70
"Saving a Device State", page 18

**RST*
Reset

Application: GPIB | RS-232 | MINI

Format: `*RST`

Description: The ***RST** command executes a soft reset of the GP700. All component modules are returned to the states given in Table 6-6.

Example: The following Visual Basic command executes a soft reset.

```
CALL Send(0, 3, "*RST", NLend)
```

See Also: Table 6-6, "Soft Reset State", page 67

S

S-Type Module Channel Select

Application: GPIB | RS-232 | MINI

Format: S<*module*> <*state*>

Description: The **S** command sets the output channel of two-position switch module number *module* to output *state*. Table 6-7 defines the actuation styles of all two-position switch configurations. A *state* value of "1" or "OFF" sets S-type modules to the positions given under "State One". A value of "2" or "ON" sets S-type modules to the positions given under "State Two". Do not send switching commands to busy modules.

Table 6-7: *Two-Position Switch Module Actuation*

Configuration	State One	State Two
On/Off	off	on
1×2	channel 1	channel 2
2×2[a]	bypass	insert

a. The actuation style of 2×2 switch modules may vary from device to device. Refer to your configuration diagram to verify the actuation style of your device.

Example: The following Visual Basic command moves two-position switch module S1 to output state 2.

```
CALL Send(0, 3, "S1 2", NLend)
```

See Also: "Command Timing", page 39

S0

S-Type Module All Call

Application: GPIB | RS-232 | MINI

Format: S0 <*state*>

Description: The **S0** command synchronously sets all two-position switch modules to output *state*. Table 6-7 defines the actuation styles of all two-position switch configurations. A *state* value of "1" or "OFF" sets S-type modules to the positions given under table heading "State One". A value of "2" or "ON" sets S-type modules to the positions given under table heading "State Two".

The All Call statement allows you to use a GP700 configured with multiple S-type modules as a multiplexed switch, following a set of configuration dependent rules.

Two-position switch modules are banked within the GP700. The degree to which two-position switches can be synchronized depends upon your banking configuration. Bank 1 can handle two S-type modules. Banks 2 and 3 can handle up to eight modules each. Each bank is controlled by a separate subprocessor. True synchronization can be achieved only between and within Banks 2 and 3. Between Bank 1 modules, and between Bank 1 and Banks 2 and 3, there is a relative delay in starting switch motion on the order of microseconds.

Table 6-8 is a partial list of typical banking configurations for the GP700. It is possible to have more than 16 two-position switches in a single GP700.

Table 6-8: *Typical Two-Position Switch Module Banking Configurations*

Total	Bank 1		Bank 2		Bank 3	
Quantity	**Quantity**	**Modules**	**Quantity**	**Modules**	**Quantity**	**Modules**
1 – 2	all	S1 – S2	—	—	—	—
3 – 8	—	—	all	S1 – S8	—	—
3 – 10	2	S1 – S2	balance	S3 – S10	—	—
11 – 16	—	—	8	S1 – S8	balance	S9 – S16

The GP700 synchronizes switch operation at the level of the main on-board processor, simultaneously sending switch commands to the subprocessors that control each bank of S-type modules.

The **S0** command is available only when controlling or downloading programs remotely. The command is not available from the front panel. Do not send switching commands to busy modules.

Example: The following Visual Basic command sets all two-position switch modules to output state 1.

```
CALL Send(0, 3, "S0 1", NLend)
```

See Also: "Command Timing", page 39
Table 6-7, "Two-Position Switch Module Actuation", page 68

S?

S-Type Module Channel Query

Application: GPIB RS-232 MINI

Format: `S<module>?`

Description: The **S?** command places the current output state of two-position switch module number *module* into the output queue. Table 6-7 defines the actuation styles of all two-position switch configurations. A returned value of "1" indicates that the module is set to the position given under table heading "State One". A value of "2" indicates that the module is set to the position given under table heading "State Two".

Example: The following Visual Basic code places the output channel of two-position switch module S1 into the output queue, and then reads the output queue into `buffer$`.

```
CALL Send(0, 3, "S1?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "2" would indicate that S-type module S1 is set to state 2.

See Also: Table 6-7, "Two-Position Switch Module Actuation", page 68

**SAV*

Save Device State

Application: GPIB RS-232 MINI

Format: `*SAV <register>`

Description: The ***SAV** command saves the current state of all modules into memory register number *register*. User-configurable memory registers 1 – 9 can store the channel, center-wavelength, and attenuation settings of all component modules.

While it is possible to save a partial switch state from the front panel, it is not possible to remotely save a partial switch state. To recall a device state, use the Recall Switch State command (***RCL**).

Example: The following Visual Basic command saves the current state of all modules into memory register 1.

```
CALL Send(0, 3, "*SAV 1", NLend)
```

See Also: ***RCL**, page 66
"Saving a Device State", page 18

SERIAL:CLOSE

Close RS-232 Interface

Application: GPIB **RS-232** MINI

Format: `SERIAL:CLOSE`

Description: The **SERIAL:CLOSE** command unlocks front-panel operation, reversing the effect of the Open RS-232 Interface command (**SERIAL:OPEN**). The **SERIAL:CLOSE** command is equivalent to pressing the **LOCAL** key on the front panel.

Example: The following Visual Basic command unlocks the front panel.

```
Comm1.Output = "SERIAL:CLOSE" + Chr$(10)
```

See Also: **SERIAL:OPEN**, page 71

SERIAL:OPEN

Open RS-232 Interface

Application: GPIB **RS-232** MINI

Format: `SERIAL:OPEN`

Description: Upon receiving the **SERIAL:OPEN** command, the GP700 enters remote mode. There is no indication of this on the display. To unlock the front panel, press the **LOCAL** key or send the Close RS-232 Interface command (**SERIAL:CLOSE**).

Example: The following Visual Basic command locks the front panel.

```
Comm1.Output = "SERIAL:OPEN" + Chr$(10)
```

See Also: **SERIAL:CLOSE**, page 71

**SRE*

Service Request Enable

Application: **GPIB** RS-232 MINI

Format: `*SRE <mask>`

Description: The ***SRE** command assigns the value of *mask* to the Service Request Enable Register. Every bit in the Service Request Enable Register corresponds to a specific event. To enable an event bit, set the bit to 1. Set masked bits to 0.

When an enabled event takes place the GP700 lights the SRQ front-panel indicator, sets the RQS bit in the Status Byte Register, and generates a service request by asserting the SRQ bus line. You can determine which device generated the service request by serial polling or by sending a Status Byte Query command (***STB?**) to every device on the bus. To determine which event took place, use the ***STB?** command. To read the current mask, use the Service Request Enable Query command (***SRE?**).

Enabling or masking an event does not affect whether an event is logged in the Status Byte. It affects only whether you are informed by service request that an event has taken place. It is possible to verify that an event has taken place without defining a Service Request Enable Register mask by repeatedly checking the Status Byte using the ***STB?** command.

***Table 6-9:** Service Request Enable Register*

Name	Description	Bit	Decimal
DIO1	GP700 is Busy	0	1
DIO2	Mini-Program End Encountered	1	2
DIO3	Mini-Program Wait Encountered	2	4
DIO4	Unused	3	8
MAV	Message Available	4	16
ESB	Event Summary Bit	5	32
RQS	Request Service	6	64
DIO8	Unused	7	128

Example: The following Visual Basic command enables all available bits in the status byte (1 + 2 + 4 + 16 + 32 + 64 = 119).

```
CALL Send(0, 3, "*SRE 119", NLend)
```

See Also: ***SRE?**, page 72
***STB?**, page 73

**SRE?*

Service Request Enable Query

Application: GPIB | RS-232 | MINI

Format: `*SRE?`

Description: The ***SRE?** command places the contents of the Service Request Enable Register into the output queue. Every bit in the Service Request Enable Register corresponds to a specific event. Enabled event bits are set to 1. Masked bits are set to 0.

When an enabled event takes place the GP700 lights the SRQ front-panel indicator, sets the RQS bit in the Status Byte, and generates a service request by asserting the SRQ bus line. You can determine which device generated the service request by serial polling or by sending a Status Byte Query command (***STB?**) to every device on the bus. To determine which event took place, use the ***STB?** command. To set the current mask, use the Service Request Enable command (***SRE**).

Example: The following Visual Basic code places the contents of the Service Request Enable Register into the output queue, and reads the output queue into `buffer$`.

```
CALL Send(0, 3, "*SRE?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "18" (binary 0001 0010) would indicate that the Message Available and Mini-Program End Encountered events are enabled to generate service requests.

See Also: ***SRE**, page 71
***STB?**, page 73
Table 6-9, "Service Request Enable Register", page 72

**STB?*

Status Byte Query

Application: GPIB | RS-232 | MINI

Format: `*STB?`

Description: The ***STB?** command places the contents of the Status Byte Register into the output queue. The Status Byte is defined in Table 6-10.

***Table 6-10:** Status Byte Register*

Name	Event	Bit	Decimal
DIO1	0 - No movement occurring. 1 - A module is currently moving.	0	1
DIO2	0 - Mini-program end not encountered. 1 - Mini-program end encountered.	1	2
DIO3	0 - Mini-program wait not encountered 1 - Mini-program wait encountered	2	4
DIO4	unused	3	8
MAV	0 - No entries in the output queue. 1 - There is an entry in the output queue.	4	16
ESB	0 - No standard event occurred. 1 - A standard event occurred.	5	32
RQS	0 - No enabled service request occurred. 1 - An enabled service request occurred.	6	64
DIO8	unused	7	128

The ***STB?** command can be used to synchronize switch commands so that commands are sent only when the GP700 is not busy. Upon receipt of a command, there is a short processing latency before DIO1 transitions from 0 to 1, indicating busy status. The completion of all pending operations is indicated by a transition of DIO1 from 1 to 0. To capture this transition, directly monitor the Status Byte with the ***STB?** command. To prevent the busy bit (DIO1) from asserting a service request, use the ***SRE** command to mask bit 1 of the Status Byte.

Example: The following Visual Basic code places the contents of the Status Byte into the output queue, and reads the output queue into `buffer$`.

```
CALL Send(0, 3, "*STB?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of 18 (binary 0001 0010) would indicate that there is an entry in the output queue and that the GP700 had encountered a mini-program end since the previous ***STB?** command.

See Also: ***SRE**, page 71

SYSTem:CONFig?

System Configuration Query

Application: GPIB | RS-232 | MINI

Format: `SYST:CONF?`

Description: The **SYST:CONF?** command places the configuration of all GP700 component modules into the output queue. The information is returned as a comma-separated list of module configurations on one line. The command returns

MATRIX INPUT<*inputs*> OUTPUT<*outputs*>,
M01 A<*outputs*> B<*inputs*>, ... ,
M<*n*> A<*outputs*> B<*inputs*>,
A01 <*max_attenuation*>, ... ,
A<*i*> <*max_attenuation*>,
F01 MIN<*min_cwl*> MAX<*max_cwl*>, ... ,
F<*j*> MIN<*min_cwl*> MAX<*max_cwl*>,
S<*k*>

where *inputs* is the number of input channels, *outputs* is the number of output channels, *n* is the number of installed M-type modules, *max_attenuation* is the maximum attenuation in dB, *i* is the number of installed A-type modules, *min_cwl* is the minimum center wavelength in nm, *max_cwl* is the maximum center wavelength in nm, *j* is the number of installed F-type modules, and *k* is the number of installed S-type modules. The returned string always ends with "S<*k*>", even when *k* = 0.

Example: The following Visual Basic code places the GP700 configuration string into the output queue, and the reads the output queue into `buffer$`.

```
CALL Send(0, 3, "SYST:CONF?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "`MATRIX INPUT04 OUTPUT08, M01 A017 B001, M02 A017 B003, F01 MIN1555.60 MAX1586.32, S04`" would indicate that the GP700 is configured with:

- one 4×8 matrix switch module
- one 1×17 multi-channel switch module M1
- one 3×17 multi-channel switch module M2
- one filter module with a center-wavelength range of 1555.60–1586.32nm.
- four two-position switch modules.

SYSTem:DATE

Set System Date

Application: GPIB RS-232 MINI

Format: `SYST:DATE <year>, <month>, <day>`

Description: The **SYST:DATE** command sets the date function of the GP700's internal clock. The *year* can be any integer in the range 1970 – 2069, the *month* any integer in the range 1–12, and the *day* any integer in the range 1 – 31.

Example: The following Visual Basic code sets the GP700's internal clock to July 4, 2001.

```
CALL Send(0, 3, "SYST:DATE 2001, 6, 4", NLend)
```

SYSTem:DATE?

System Date Query

Application: GPIB RS-232 MINI

Format: `SYST:DATE?`

Description: The **SYST:DATE?** command queries the system clock and places the resulting date into the output queue. The returned date string is

<year>, <month>, <day>

where *year* is the four-digit year, *month* is the one- or two-digit month, and *day* is the one- or two-digit day.

Example: The following Visual Basic code places the GP700's current date into the output queue, and then reads the output queue into `buffer$`.

```
CALL Send(0, 3, "SYST:DATE?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "`1999, 12, 28`" would indicate that the GP700 date is set to December 28, 1999.

SYSTem:ERRor?

System Error Query

Application: GPIB RS-232 MN

Format: `SYST:ERR?`

Description: The **SYST:ERR?** command reads and removes one error message from the GP700 error queue. The result is placed into the output queue. The command returns

<error_code>, <error_message>

where *error_code* is the error code number and *error_message* is a message string describing the type of error. If the queue contains no errors, the command will return "`+0, No Error`".

Example: The following Visual Basic code copies the first error-queue message into the output queue, clears the message from the error queue, and reads the output queue into `buffer$`.

```
CALL Send(0, 3, "SYST:ERR?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "`-224, Illegal parameter value`" would most likely indicate an attempt to address a non-existent module or out-of-range module setting.

See Also: "Remote Error Codes", page 83

SYSTem:TIME

Set System Time

Application: GPIB RS-232 MINI

Format: `SYST:TIME <hour>, <minute>, <second>`

Description: The **SYST:TIME** command sets the time function of the GP700's internal clock. The *hour* can be any integer in the range 0–23, the *minute* any integer in the range 0–59, and the *second* any integer in the range 0–59.

Example: The following Visual Basic command sets the GP700's internal clock to 5:15:20 PM.

```
CALL Send(0, 3, "SYST:TIME 17, 15, 20", NLend)
```

SYSTem:TIME?

System Time Query

Application: GPIB RS-232 MINI

Format: `SYST:TIME?`

Description: The **SYST:TIME?** command queries the system clock, and places the resulting time of day into the output queue. The returned string is

<hour>, <minute>, <second>

where *hour* is the one- or two-digit hour (24-hour clock), *minute* is the one- or two-digit minute, and *second* is the one- or two-digit second.

Example: The following Visual Basic code places the GP700's current time into the output queue, and then reads the output queue into `buffer$`.

```
CALL Send(0, 3, "SYST:TIME?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, a `buffer$` value of "`14, 30, 1`" would indicate that the GP700 clock is set to 2:30:01 PM.

TOGS
Toggle S-Type Module

Application: GPIB RS-232 MINI

Format: `TOGS<module>`

Description: The **TOGS** command toggles the state of two-position switch module number *module*. Do not send switching commands to busy modules.

Example: The following Visual Basic command toggles S-type module S1.

```
CALL Send(0, 3, "TOGS1", NLend)
```

See Also: "Command Timing", page 39
Table 6-7, "Two-Position Switch Module Actuation", page 68

**TRG*
Trigger

Application: GPIB RS-232 MINI

Format: `*TRG`

Description: The ***TRG** command sends a trigger message. The GP700 interprets trigger messages only in the context of mini-programs.

When a GP700 mini-program is waiting for a trigger, the front-panel display prompt shows "`WAIT`" followed by a description indicating the type of trigger that will end the wait. For example, the display prompt shows "`WAIT GPIB`" when a mini-program is waiting for a trigger transmitted via the GPIB interface. The GP700 also sets the DIO3 bit in the Status Byte Register to 1.

Example: The following Visual Basic command sends a trigger message to the GP700.

```
CALL Send(0, 3, "*TRG", NLend)
```

See Also: "Waiting for a Trigger", page 27
Table 6-10, "Status Byte Register", page 73
WAIT, page 80

*TST?

Self-Test Query

Application: GPIB RS-232 MINI

Format: `*TST?`

Description: The ***TST** command places a "+0" into the output queue, verifying the GP700's ability to receive and respond to commands.

Example: The following Visual Basic code places a "+0" into the output queue, and reads the output queue into `buffer$`.

```
CALL Send(0, 3, "*TST?", NLend)
CALL Receive(0, 3, buffer$, NLend)
```

Following execution, if `buffer$` does not equal "+0" an interface error has occurred.

See Also: "Servicing and Troubleshooting", page 81

*WAI

Wait-To-Continue

Application: GPIB RS-232 MINI

Format: `*WAI`

Description: The ***WAI** command causes the GP700 to wait until all pending operations are complete before executing the next command. Use the ***WAI** command to prevent command collisions.

Example: The following Visual Basic code causes the GP700 to wait until not busy.

```
CALL Send(0, 3, "*WAI", NLend)
```

WAIT

Wait For Trigger Input

Application: MINI

Format: `WAIT <type>`

Description: The **WAIT** instruction places a wait in a mini-program. The program waits until it receives the trigger input specified by *type*. There are four wait types and triggers. A wait of *type* "G" responds to a ***TRG** command via the GPIB interface or a GPIB Group Execute Trigger (**GET**) message. For more information about the **GET** message refer to your GPIB controller documentation. A wait of *type* "K" responds to a keypad event. A wait of *type* "R" responds to a ***TRG** command via the RS-232 interface. A wait of *type* "X" responds to a signal via the rear panel external trigger input.

***Table 6-11:** Mini-Program Wait Options*

Wait Type	Description
G	GPIB Interface Wait: the mini-program pauses until it receives a Trigger command (***TRG**) via the GPIB interface or a GPIB **GET** message.
K	Keyboard Wait: the mini-program pauses until the user presses **ENTER**. This feature can be used to pause your program while a user notes intermediate results.
R	RS-232 Interface Wait: the mini-program pauses until it receives a Trigger command (***TRG**) via the RS-232 interface.
X	External Trigger Wait: the mini-program pauses until it receives a trigger signal via the external trigger port. The trigger type and polarity must match the configuration of the external trigger port.

When the GP700 encounters a mini-program wait, the DIO3 bit in the Status Byte Register is set to 1. The front-panel display shows which type of wait has halted a mini-program. For example, if the wait message is "WAIT ENTER KEY", the user must press **ENTER** to continue the mini-program. Note that if the mini-program was executed remotely using the **PROG:EXEC** command, the GP700 may be in remote mode. In this case, press the **LOCAL** key to enable keypad input, then press the **ENTER** key to continue the mini-program.

Example: The following Visual Basic command appends a Wait For Trigger Input command to mini-program 1. The instruction causes the mini-program to halt execution and wait for a keyboard event.

```
CALL Send(0, 3, "PROG1:NEWL 'WAIT K'", NLend)
```

See Also: "Local, Remote, and Local Lockout Modes", page 39
Table 6-10, "Status Byte Register", page 73
PROGram:EXECute, page 65
***TRG**, page 78

Chapter 7
Servicing and Troubleshooting

This chapter contains information about:

- performing a functional test
- investigating common problems
- interpreting remote error codes
- improving repeatability
- contacting DiCon

Performing a Functional Test

The GP700 performs a self-test upon power-up. This test and the Self-Test Query remote command (see ***TST?**, page 79) are the only forms of self-test available. The ERR front-panel indicator will light if the power-up self-test finds an error. In this case, cycle power to repeat the self-test. If the problem persists, call DiCon for further instructions.

Investigating Common Problems

Table 7-1 presents solutions to problems you might encounter while setting up and using the GP700. If after following the troubleshooting guide you are unable to correct the problem please contact DiCon.

Table 7-1: *Troubleshooting Guide*

Problem	Possible Solution
No light. High insertion loss.	Connectors are dirty, or your test jumper is broken (see "Handling Fiberoptic Cables and Connectors", page 2).
All channels show high insertion loss.	The input or common connector is dirty, or a jumper to one of these connectors is dirty (see "Handling Fiberoptic Cables and Connectors", page 2).
A channel shows high insertion loss.	The connector is dirty, or the jumper to the connector is dirty. (see "Handling Fiberoptic Cables and Connectors", page 2).
I need to improve optical repeatability.	See "Improving Repeatability", page 84.

Table 7-1: *Troubleshooting Guide (Continued)*

Problem	Possible Solution
GP700 display does not turn on.	Check power connection.
Error indicator lights at power up.	Contact DiCon.
Module won't change setting.	If you are operating via the front panel, verify that the correct module is selected (blinking). See "Front-Panel Operation", page 15.
Recalling a state from memory moves only a subset of modules.	You have recalled a partial device state. See "Saving a Device State", page 18.
Can't get out of mini-program record mode.	Press the **PROG** softkey, then **EXIT** or **SAVE**.
Mini-program continually errors.	Likely errors include a mismatched program loop, too many nested program loops, or a relative setting that results in an out-of-range position. Re-enter the program taking care to match loop delimiters. Do not nest loops more than four deep.
Mini-program halts, shows "`WAIT`" statement.	The mini-program is waiting for a trigger. The appropriate trigger type is displayed in the prompt line. See "Waiting for a Trigger", page 27.
Mini-program halts, shows "`WAIT TRIGGER IN`" but does not respond to input triggers.	Verify that the external trigger configuration matches the actual trigger input. See "Configuring the Input/Output Ports", page 19.
Keys don't work, keypad is locked.	If you are remotely addressing the GP700 via GPIB or RS-232, press the **LOCAL** key. Note that the GP700 will return to remote mode as soon as it receives another remote command. See "Local, Remote, and Local Lockout Modes", page 39.
Modules are moving but display does not update.	The Turn Off Display command has been sent remotely. See **DISPlay:OFF**, page 50.
Can't address the GP700 remotely via GPIB.	When the GP700 is successfully addressed via GPIB, the RMT front-panel indicator lights. If the indicator does not light, verify that the GPIB address is set properly, that the address is unique on the bus, and that you are using the proper command terminator. See "Remote Operation", page 31. Try addressing another device on the same bus. If you are unable to address the other device, there is something wrong with your interface board or configuration. If you are able to address the other device but still can't address the GP700, contact DiCon.
Can't address the GP700 remotely via RS-232.	The RMT front-panel indicator does not light when the GP700 is successfully addressed via RS-232. If the GP700 does not respond appropriately to your commands, verify that the baud rate is set properly and that you are using the proper command terminator. See "Remote Operation", page 31. Be careful to send the entire command in one packet. Try attaching a different device to the interface. If you are unable to address the other device there is something wrong with your interface board or configuration. If you are able to address the other device but still can't address the GP700, contact DiCon.
Continually get remote errors when sending multiple commands.	This is typically caused by bus collisions within the GP700 when commands are sent too quickly (see "Command Timing", page 39). Don't send commands to busy modules.
May I open the switch case?	No, this voids the warranty. Contact DiCon Fiberoptics for permission prior to attempting to open the GP700.

Interpreting Remote Error Codes

The GP700 error codes and descriptions are listed in Table 7-2.

Table 7-2: *Remote Error Codes*

Code	Description
-102	Syntax error. Required parameter list missing, incorrect separator between multiple parameters, or extra characters detected after valid command.
-104	Syntax error. Unrecognized data type received. Improperly formed floating point number sent.
-105	Syntax error. Bad or missing string delimiter in string data. Mismatched quotes.
-106	Syntax error. Remote command ended unexpectedly. All the bytes in the command were not sent.
-109	Syntax error. Required parameter missing from command.
-112	Command too long. GP700 input buffer exceeded.
-113	Unrecognized command received. Command header is not recognized.
-121	Syntax error. Improperly formed digit received. Parameter expected to be a number contains non-numeric data.
-202	Error buffer became out of sequence. Call DiCon if error recurs.
-222	Out of range parameter received. Attempt to save or recall a non-existent or out of range state.
-223	Attempt to send string that is too long. Attempt to send an Append New Line to Mini-Program command (**PROG:NEWL**) that is too long.
-224	Illegal parameter value received. Attempt to move a non-existent module, to move a non-existent port on a module, to move to a non-existent channel, to attempt to read, write, or run to a non-existent mini-program, to set an invalid time or date, or to configure an I/O port improperly.
-230	Requested mini-program does not exist. An attempt to run a valid mini-program number which contains no mini-program, or an attempt to run a mini-program while currently executing a mini-program.
-231	Mini-program too large. Mini-program buffer exceeded.
-240	Out of memory trying to do mini-program read back.
-350	Error queue overflow. Use the System Error Query Command (**SYST:ERR**) to clear the remote error queue.
-444	Query unterminated. An attempt to read the GPIB output queue/buffer when no data was there.
-445	Query interrupted. The device received a new query command before the controller read data from a previous query command.
-447	String too long. The device received a command string that overflowed the internal GPIB receive buffer.
-446	Internal error: invalid token from GPIB driver. Call DiCon if error recurs.
401	Incorrect checksum received from module. You may be operating in a noisy electrical environment. Call DiCon if error recurs.
403	Tried talking to busy module. See "Command Timing", page 39.
500	The requested channel number cannot be transmitted to the M-type module. Call DiCon if error recurs.
600	Transmission time-out talking to module. Call DiCon if error recurs.
601	Received time-out talking to module. Call DiCon if error recurs.
602	Incorrect response received from module. You may be operating in a noisy electrical environment. Call DiCon if error recurs.

Table 7-2: Remote Error Codes

Code	Description
603	Incorrect number of bytes received from module. You may be operating in a noisy electrical environment. Call DiCon if error recurs.
604	No end frame byte received from module. You may be operating in a noisy electrical environment. Call DiCon if error recurs.
700	Requested A-, F-, I-, or M-type module does not exist. Call DiCon if error recurs.
701	Requested S-type module does not exist. Call DiCon if error recurs.
901	Tried to run mini-program while one is already running.
903	Mini-program command too long or no mini-program end encountered before the end of mini-program memory.
904	Encountered an End Loop instruction without a corresponding Begin Loop instruction, or loops are nested more than four deep.
905	User requested that a mini-program stop running.
1201	A-, F-, I-, M-, or S-type module failed to respond to a request. Call DiCon if error recurs.
1203	A-, F-, I-, M-, or S-type module failed to move to the desired position due to an error in the module. Call DiCon if error recurs.
1204	A-, F-, I-, M-, or S-type module could not be found at start-up and is not available for use. Call DiCon if error recurs.
1400	Trying to move a S-type module while it is already moving. See "Command Timing", page 39.
1500	The requested attenuation level cannot be transmitted to the A-type module. Call DiCon if error recurs.
1600	The requested center-wavelength value cannot be transmitted to the F-type module. Call DiCon if error recurs.

Improving Repeatability

The following sections discuss the most successful strategies for optimizing system repeatability.

Warm Up Source, Detector, and Switch

Before taking measurements the optical source, detector, and switch should be connected and powered on for more than two hours. During this time the switch should be continuously cycled. Warming up stabilizes the test equipment temperature. Cycling the switch minimizes fiber stresses.

Use An Ultra-Stable Light Source

Use only the most stable light source available. If you are measuring only insertion loss, an LED source may be used. LED sources are more stable than laser diodes. When using a laser diode, avoid coherence problems by using an FP laser rather than a DFB laser. Finally, if your source is stable only over the short term, monitor source drift using multiple reference channels (see "Monitor Source Drift", page 86).

Avoid Source-Destabilizing Techniques

When using dual wavelength sources (e.g. 1310 and 1550nm), ensure that both optical sources remain powered at all times. Some single-port dual-wavelength sources apply power to only one source at a time. When switching between wavelengths, power is cycled to the source, and the entire warm-up period must be repeated before stable measurements are possible. To avoid these complications, use two separate sources and a 1×2 switch to change wavelengths.

Minimize Back-Reflection

Particularly when working with laser diodes, it is very important to minimize reflected signals. Signals reflected into a laser source will destabilize the source. Lasers are sensitive to both the magnitude and the polarization state of the reflected light. To minimize reflections use:

- *An Isolator in front of the laser source.* Isolators allow light to pass in only one direction. When used properly, isolators can dramatically reduce back-reflection into the source.
- *A low back-reflection attenuator in front of the source.* This method attenuates both the outgoing source signal and the incoming reflected signal. This means that the reflected signal is attenuated twice, thereby strongly reducing back-reflection into the source. Of course, your source output signal is also attenuated.
- *Low back-reflection connectors throughout the test set-up.* A dramatic reduction in back-reflection can be obtained by angle polishing the connectors in the test set-up. Reflected signals cannot propagate through these connectors. Use angle-polished connectors at the source, the detector, and at any open fiber port to reduce system back-reflection. DiCon recommends a minimum angle of 8°.

Use an Ultra-Stable Detector

Detectors tend to be temperature sensitive. To reduce the possibility of corrupting your data due to detector temperature variations use only temperature-controlled detectors. These are frequently called TE detectors, where TE is short for thermo-electrically cooled.

If power level changes are being made with a single fiber and a single detector, choose the detector with the best linearity (we recommend the HP 815323A). If power level comparisons are being made between multiple detectors, choose detectors with the best total accuracy.

When using the detector, allow enough averaging time to ensure that short term variations get averaged out. Set the averaging time such that when monitoring a typical signal no variation is measured. This can be particularly important when making low-light measurements such as back-reflection, where even the slightest variation can corrupt your data. In such cases, DiCon recommends you use a detector that offers both high sensitivity to low-light levels, and the ability to set detector averaging times.

Use Low-PDL Components

Eliminate sources of polarization dependent loss (PDL) within your test system by using low-PDL components. The polarization state of the light within your test system typically varies continuously. Reducing the PDL of each component within your system will reduce the effect of this variation upon your measurements.

Use a Detector With Low Polarization Sensitivity

Many of the detectors currently on the market are sensitive to polarization. Polarization-sensitive detectors will indicate different power levels for two equivalent signals differing only in polarization state but not in optical power. To avoid corrupting your data with variations due to polarization sensitivity use a detector with low polarization sensitivity. Note that low-PDL detectors may not have sufficient dynamic range to measure back-reflection.

Monitor Source Drift

For long-term testing it may be necessary to actively monitor source output and to normalize your measurements over time. This is typically accomplished using a 1:2 or 2:2 splitter with a second detector, or by using a switch channel as a reference channel which bypasses all devices under test. You can enhance the accuracy of the reference measurement by using multiple switch reference channels, and then averaging the reference values at each measurement cycle. Bellcore, for example, recommends using four averaged switch channels as a reference. With power monitoring in place, normalize each measurement taken to the corresponding reference power.

Maintain Stable Temperature

Temperature fluctuations can disrupt the stability of sources, detectors, fiberoptic couplers, fiberoptic switches, and other components. Temperature changes should be limited to ±0.5°C when attempting to make repeatability measurements accurate on the order of 0.01 dB. This can be accomplished by mounting your test system in a 19" temperature-controlled rack. Contact DiCon for names of vendors of suitable rack systems.

Use Single-Channel Sequential Switching

Matrix and multi-channel switch modules are more repeatable when switching in a consistent sequence. During testing, it is recommended that testing proceed in single channel, forward increments (e.g. 1, 2, 3, 4, … , N, Reset, 1, 2, 3, 4, … , N). The short term repeatability of a 1×N Switch using a single-channel switching sequence is ±0.005 dB. Using a random switching sequence, repeatability is ±0.025 dB.

Minimize Connectors and Couplers

Connectors, couplers, splices, and other components contribute to measurement uncertainty. To minimize this uncertainty, fusion splice the device under test into the test system, and minimize the use of connectors and other components. Note that fusion splices are not 100% stable, and have been observed to cause instantaneous spikes in optical measurements, possibly due to local stresses in the fusion splice junction.

Eliminate Fiber Stress

Particularly when using a laser source, slight stresses or vibrations in the measurement system will cause measurement variations. This phenomenon can be demonstrated by connecting the source directly to a detector with a fiber jumper. While monitoring power variation, move the fiber jumper slightly. You should observe a change in output power. To minimize these effects, fix all loose fibers to the measurement table and avoid all vibration of the measurement set-up. Handle fiber with care (see "Handling Fiberoptic Cables and Connectors", page 2).

In addition, localized stress points in the cable layout can cause stress build-up. This stress can degrade performance by, for example, enhancing polarization sensitivity or causing cable losses to creep over time. To avoid these problems, use your eyes and fingers to trace the path of all external cable pigtails. Starting from the source, work your way along each channel looking for localized stress points and too-small bend radii. In particular you should watch for the following:

- *Excessive bending of the fibers.* Any local bends in the cable less then 5cm (2in.) in diameter can cause high losses and instabilities over time. These effects are likely to be more pronounced at longer wavelengths, such as 1550 nm. Small bumps in the cable layout due to edges of table tops or cable race-tracks can create similar effects. Some cable constructions (tight buffer) may allow stress on the jacket to be transferred to the fiber itself.
- *Tight tie wraps.* Over time, overly tight tie wraps will dent fiber jackets and eventually transfer those stresses through to the fiber. If you must use tie wraps, make sure that they are loose enough such that they can easily be jiggled back-and-forth with your fingers. If tie wraps require any force to move around or if you can see dents in the cable jacket, the tie wraps are too tight and should be loosened or removed.
- *Cables laid out over sharp edges.* If individual cables or bundles of cables are laid over a sharp edge, cable denting can create localized stress in the fiber in much the same way as a too-tight tie wrap. The best way to avoid this is to lay all cables flat, even if this means that several cables will be laid one atop another. If you cannot avoid crossing a sharp edge, place a layer of high-density foam rubber between the cables and the sharp edge to distribute the effects of the edge over a larger surface.
- *Vertical cable sections.* The problem with vertical cable sections is that the full weight of the hanging cable will create stresses at any point where the cable is tied down, or where it is bent at 90 degrees to go horizontal again. Reduce the magnitude of the resulting stresses by using wide tape frequently along the cables length for support. The use of wide tape will help to prevent the build-up of localized stresses.
- *Stresses near the bulkhead.* Allow adequate space behind the GP700 to stow excess fiber without weighing down cables at the bulkhead. In addition, never use the bulkhead feedthroughs or adapters to support fiber bundles. Laying fiber bundles on top of bulkhead pigtails will create localized stress and can potentially damage the fiber.

Eliminate Measurement Aberrations

Use mathematical techniques to filter measurement noise. One common technique is to discard the first measurement, and then take multiple measurements accepting the data only after four consistent measurements are recorded. This will filter out any data spikes caused by fiber stress change, polarization effects, etc. Also, since the detector goes dark (does not receive light) for a short period of time as the switch moves to a new position, it may take a finite amount of time for the detector output to stabilize at a new light level. For this reason it may be necessary to discard the first reading coming from the detector.

Contacting DiCon Fiberoptics

DiCon Fiberoptics believes in total customer satisfaction. We want to hear from you both when things go right and when things go wrong. If you have any questions or comments about the GP700, or any other product in our line, feel free to contact us at:

Customer Support
DiCon Fiberoptics Inc.
1331 Eighth Street
Berkeley, CA 94710
Phone: (510) 528-0427
FAX: (510) 528-1519
Email: info@diconfiberoptics.com
Web: http://www.diconfiberoptics.com

On the last page of this manual you will find a Comments and Suggestions Survey Form. Please feel free to fill it in and send/Fax us your comments at any time.

Appendix A
Specifications

Regulatory Conformance

The GP700 conforms to the following directives and standards:

- electromagnetic compatibility directive 89/336/EEC
- low voltage directive 73/23/EEC
- radio disturbance standard EN55022
- electrical immunity standard EN50082-1
- safety standard EN61010-1
- mounting standards IEC 297-1 and DIN 41494 Part I (rackmount chassis only)

Year 2000 Statement

The GP700 is immune to the Year 2000 problem. We have tested and verified that the product will continue to function without interruption after the change of the millennium.

The device stores the year as a two-digit value representing dates from 1970 through 2069. The front panel displays the year using these two digits. The remote System Date Query command (see **SYSTem:DATE?**, page 75) returns the year as a four-digit value in compliance with the Standard Commands for Programmable Instruments (SCPI) specification.

About Specifications

All specifications are provided for reference purposes only and are subject to change without notice. The information provided within this manual has been carefully reviewed for technical accuracy. DiCon reserves the right to correct technical or typographical errors at any time, without prior notice.

Mechanical Specifications

Table A-1: *Mechanical Specifications*

Parameter	2U Standard Rackmount	4U Standard Rackmount	4U Extended Rackmount	6U Extended Rackmount	Benchtop	Unit
width	483.0	483.0	483.0	483.0	257.4	mm
height	88.0	178.0	178.0	265.9	191.6	mm
depth	434.5	434.5	554.0	554.0	412.6	mm
weight	7.3	6.8	9.6	12.3	4.5	kg

Table A-2: *Module Weight*[a]

Parameter	Weight	Unit
A-type module	0.20	kg
F-type module	0.20	kg
I-type module (each input and output)	0.90	kg
M-type module (16 channels)	0.90	kg
M-type module (80 channels)	1.35	kg
S-type module	0.08	kg

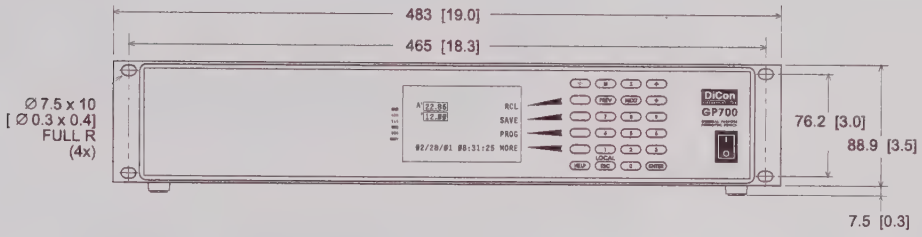
a. Module weights are approximate and subject to change at any time.

Housing Diagrams

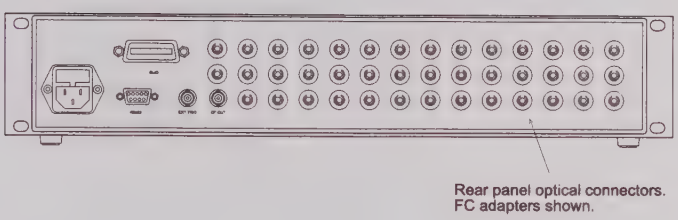
Dimensions are shown in mm [in.]. 4U and 6U rackmount chassis shipped with handles (not shown). Benchtop chassis tilt-front handle available by request. Optical interface depends upon configuration purchased.

Figure A-1: *2U Standard Rackmount Chassis*

Front



Rear



Side

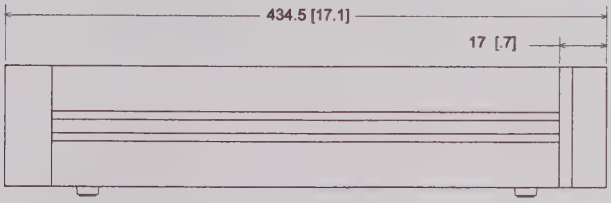
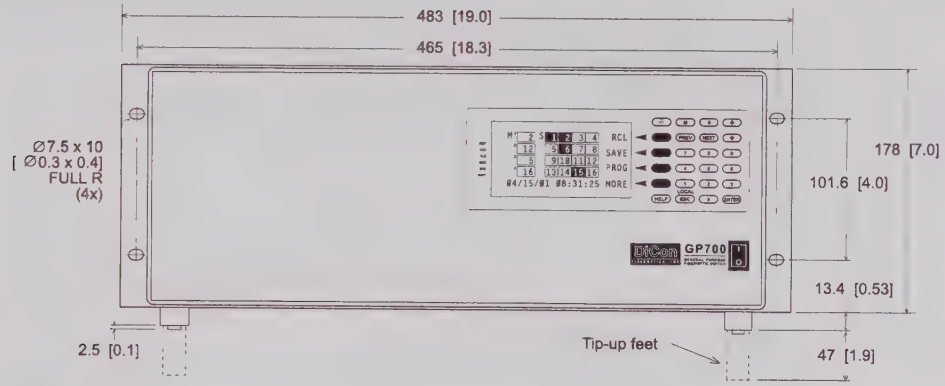
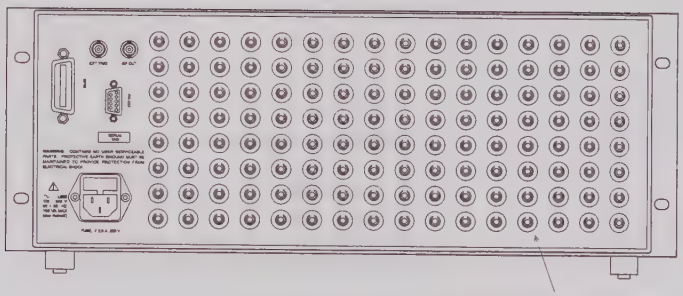


Figure A-2: *4U Standard Rackmount Chassis*

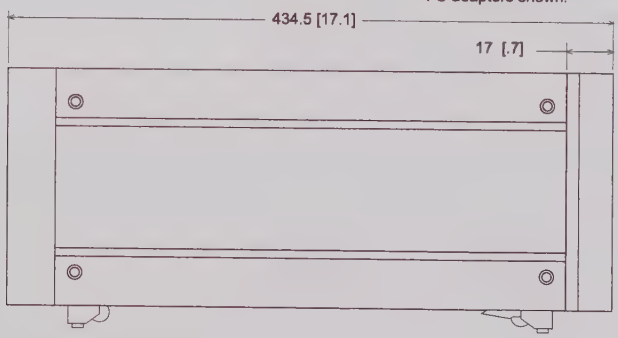
Front



Rear



Side



Note: 4U extended rackmount chassis differs only in depth (554 mm).

Figure A-3: 6U Extended Rackmount Chassis

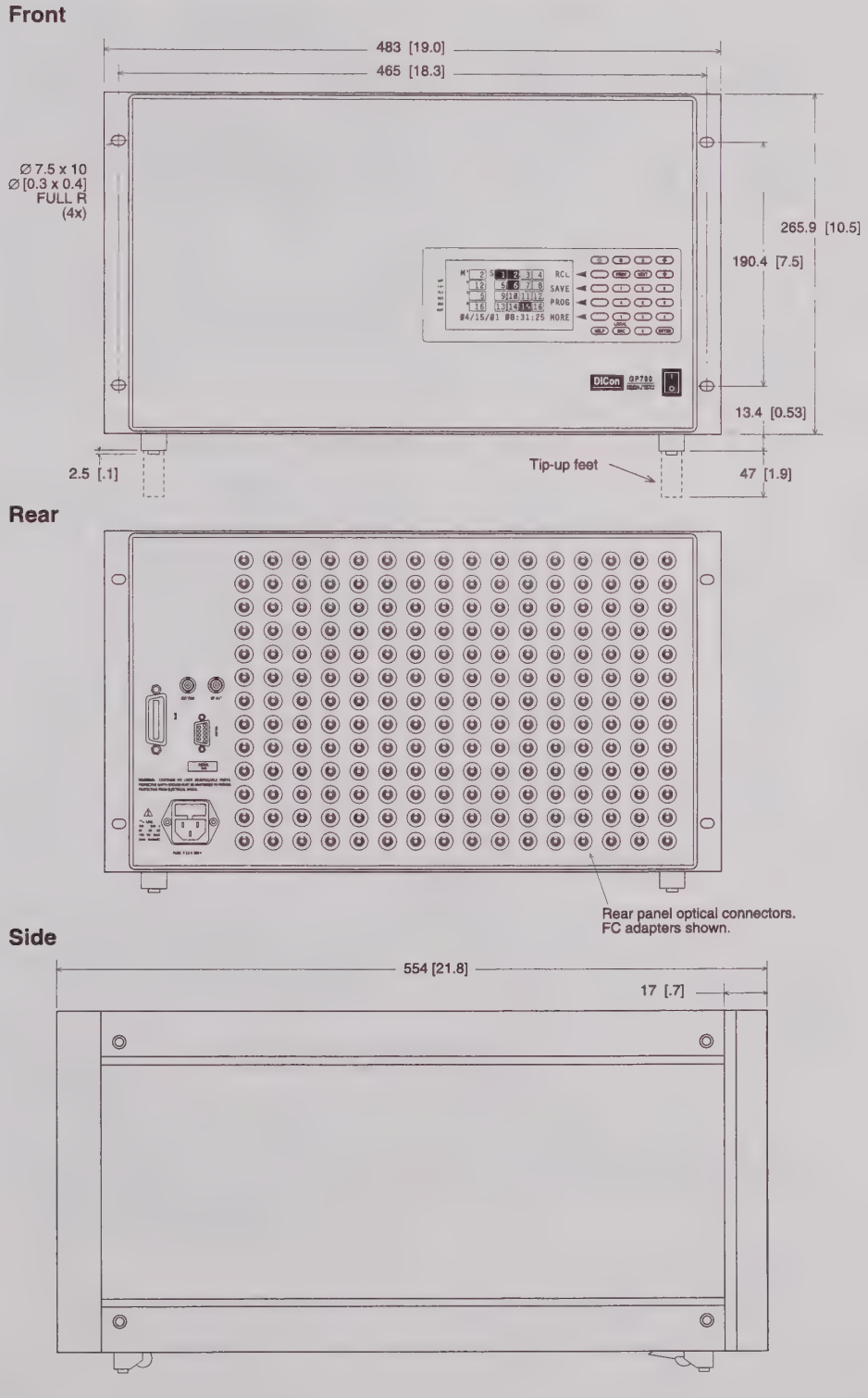
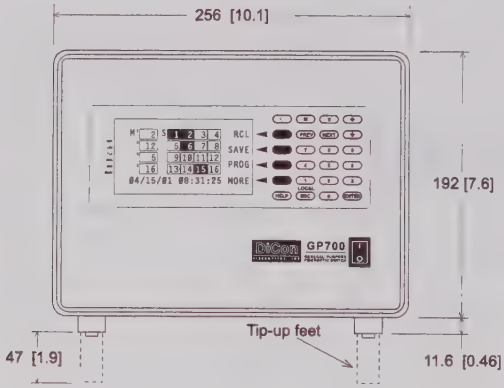
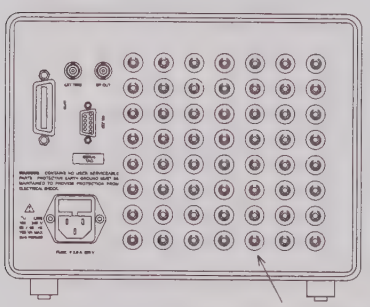


Figure A-4: Benchtop Chassis

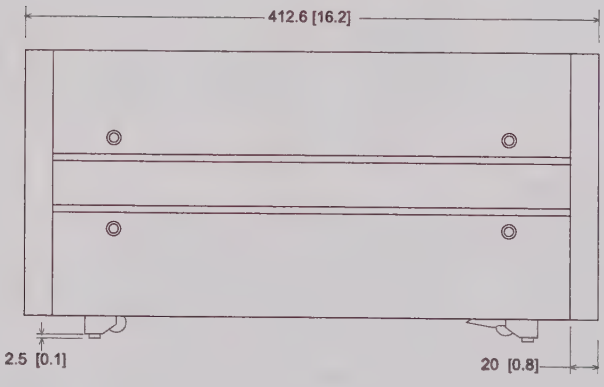
Front



Rear



Side



Electrical Specifications

Table A-3: Electrical Specifications

Parameter	Minimum	Typical	Maximum	Unit
AC Input Power Specifications				
input power	—	—	150	VA
input voltage	100	—	240	VAC
internal fuse[a]	—	2	—	A
external fuse[b]	—	—	2	A
AC input leakage current	—	—	3.5	mA
External Trigger Specifications				
logical high voltage	2	—	5	VDC
logical low voltage	0	—	0.5	VDC
input capacitance	—	—	100	pF
internal load[c]	900	1000	1100	Ω
continuous overload protection	—	±30	—	VDC
ESD overload protection	—	±10	—	kV
General Purpose Output Specifications				
logical high voltage	3	—	5	VDC
logical low voltage	0	—	0.4	VDC
switching time low-to-high	—	—	2	µs
switching time high-to-low	—	—	2	µs
output current loading[d]	—	—	50	mA
output loading restive[e]	50	—	∞	Ω
continuous overload protection[f]	—	±30	—	VDC
ESD overload protection	—	±10	—	kV
Interface Specifications				
RS-232 cable length	—	—	15.2	m
baud rate	1200 / 9600			baud
GPIB default address	3			
Internal Clock / Calendar Specifications				
accuracy	—	±1	—	min/month
lifetime	—	10	—	year

a. DiCon recommends a 2.5-A fuse for 6U-rackmount matrix switches
b. 2AG Fast Acting
c. For fast external sources a 50Ω BNC in-line terminator may be connected to reduce ringing.
d. Resistive / inductive loads.
e. Specifications met over entire range. The General Purpose Output is short-circuit protected. Upon detecting a short circuit, the output is automatically disabled. Resetting the GP700 either from the front panel or remotely re-enables the output.
f. Protected against inductive spikes when driving relays or solenoids.

Environmental Specifications

Table A-4: Environmental Specification

Parameter	Condition	Minimum	Maximum	Unit
operating temperature	—	0	40	°C
storage temperature	—	-20	60	°C
humidity	40°C / 90% RH	—	5	days

Optical Specifications

A-Type Attenuator Modules

Table A-5: Attenuator Range-Dependent Specifications[a]

Parameter	0 – 10dB	10 – 30dB	30 – 60dB	Units
resolution	0.10	0.12	0.15	dB
repeatability	0.10	0.20	0.20	dB
absolute accuracy	0.20	0.40	0.50	dB

a. Singlemode only. All specifications referenced without connectors.

Table A-6: Attenuator General Performance Specifications[a]

Parameter	Minimum	Typical	Maximum	Unit
insertion loss	—	0.6	1.5	dB
back-reflection	—	—	-50	dB
damage threshold	—	—	24	dBm
tuning time[b]	—	50	1400	ms
Busy Bit jitter[c]	—	—	±25	ms

a. Singlemode Only. All specifications referenced without connectors.
b. Tuning time is the sum of processing overhead, tuning, debounce, and completion buffer periods. See "Command Timing", page 39 for more information.
c. Measured relative to the completion of the command.

F-Type Filter Modules

Table A-7: *Filter Optical Performance*[a]

Parameter	Minimum	Typical	Maximum	Unit
tuning ranges	1290	—	1320	nm
	1535	—	1565	nm
0.5-dB bandwidths[b]	0.8, 1.5, 2, 3, or 9			nm
tuning resolution	—	0.05	—	nm
repeatability	—	±0.05	—	nm
accuracy	—	±0.3	—	nm
insertion loss[c]	—	—	1.5	dB
back-reflection	—	—	-50	dB
PDL[c d]	—	0.05	—	dB
center-wavelength stability	—	—	0.005	nm/°C
thermal stability[c]	—	—	0.005	dB/°C
tuning time[e]	50	—	1400	ms
Busy Bit jitter[f]	—	—	±25	ms

a. Singlemode only. All specifications referenced without connectors.
b. Nominal values. Not all 0.5-dB bandwidth/tuning range combinations are available with stock filter elements.
c. Measured at maximum center wavelength.
d. Maximum value may vary depending on filter type.
e. Tuning time is the sum of processing overhead, tuning, debounce, and completion buffer periods. See "Command Timing", page 39 for more information.
f. Measured relative to the completion of the command.

I-Type Matrix Switch Modules

***Table A-8:** Matrix Switch Singlemode Optical Performance*[a]

Parameter	Typical	Maximum	Unit
insertion loss[b]	1.2	2.4	dB
back-reflection[b]	-55	-50	dB
PDL	—	0.1	dB
cross-talk	-100	-80	dB
repeatability[c]	—	±0.05	dB
switching time[d]	500	750	ms
Busy Bit jitter[e]	—	±25	ms
optical jitter[f]	—	±6	ms

a. Measured at λ = 1310nm and λ = 1550nm. All specifications referenced without internal or external connectors.
b. Measured at 23±5°C.
c. 100 cycles measured at constant temperature after warm-up.
d. Switching time is the sum of the processing overhead, switching, debounce, and completion buffer periods. See "Command Timing", page 39 for more information.
e. Measured relative to the completion of the command.
f. Measured relative to the busy start point.

***Table A-9:** Matrix Switch Multimode Optical Performance*[a]

parameter	typical	Maximum	Unit
insertion loss[b]	1.2	—	dB
back-reflection[b]	-18	—	dB
cross-talk	-100	-80	dB
repeatability[c]	—	±0.05	dB
switching time[d]	500	750	ms
Busy Bit jitter[e]	—	±25	ms
optical jitter[f]	—	±6	ms

a. Measured at λ = 800nm and λ = 1310nm. All specifications referenced without internal or external connectors.
b. Measured at 23±5°C.
c. 100 cycles measured at constant temperature after warm-up.
d. Where N is the number of channels moved. Switching time is the sum of the processing overhead, switching, debounce, and completion buffer periods. See "Command Timing", page 39 for more information.
e. Measured relative to the completion of the command.
f. Measured relative to the busy start point.

M-Type Multi-Channel Switch Modules

Table A-10: *Multi-Channel Switch Singlemode Optical Performance*[a]

Parameter	Typical	Maximum	Unit
insertion loss[b]	0.6	1.2	dB
back-reflection[b]	-60	-55	dB
PDL	0.03	0.05	dB
cross talk	-100	-80	dB
repeatability (sequential)[c]	—	±0.005	dB
repeatability (random)[c]	—	±0.025	dB
switching time[d]	—	425 +12*N*	ms
Busy Bit jitter[e]	—	±25	ms
optical jitter[f]	—	±6	ms

a. Measured at λ = 1310nm and λ = 1550nm. All specifications referenced without connectors.
b. Measured at 23±5°C.
c. 100 cycles measured at constant temperature after warm-up.
d. Where *N* is the number of channels moved. Switching time is the sum of processing overhead, switching, debounce, and the completion buffer periods. See "Command Timing", page 39 for more information.
e. Measured relative to the completion of the command.
f. Measured relative to the busy start point.

Table A-11: *Multi-Channel Switch Multimode Optical Performance*[a]

Parameter	Typical	Maximum	Unit
insertion loss[b]	0.6	—	dB
back-reflection[b]	-20	—	dB
cross talk	-100	-80	dB
repeatability (sequential)[c]	—	±0.005	dB
repeatability (random)[c]	—	±0.025	dB
switching time[d]	—	425 +12*N*	ms
Busy Bit jitter[e]	—	±25	ms
optical jitter[f]	—	±6	ms

a. Measured at λ = 800nm and λ = 1310nm. All specifications referenced without connectors.
b. Measured at 23±5°C.
c. 100 cycles measured at constant temperature after warm-up.
d. Where *N* is the number of channels moved. Switching time is the sum of processing overhead, switching, debounce, and completion buffer periods. See "Command Timing", page 39 for more information.
e. Measured relative to the completion of the command.
f. Measured relative to the busy start point.

S-Type Two-Position Switch Modules

Table A-12: *Two-Position Switch Singlemode Optical Performance*[a]

Parameter	typical	maximum	Unit
insertion loss[b]	0.6	1.2	dB
back-reflection[b]	—	-55	dB
PDL	0.03	0.05	dB
cross-talk	-100	-80	dB
repeatability[c]	—	±0.005	dB
switching time[d]	—	135	ms
Busy Bit jitter[e]	—	±25	ms

a. Measured at λ = 1310nm and λ = 1550nm. All specifications referenced without connectors.
b. Measured at 23±5°C.
c. 100 cycles measured at constant temperature after warm-up.
d. Switching time is the sum of processing overhead, switching, debounce, and completion buffer periods. See "Command Timing", page 39 for more information.
e. Measured relative to the completion of the command.

Table A-13: *Two-Position Switch Multimode Optical Performance*[a]

Parameter	typical	Maximum	Unit
insertion loss[b]	0.6	—	dB
back-reflection[b]	-30	—	dB
cross talk	-100	-80	dB
repeatability[c]	—	±0.005	dB
switching time[d]	—	135	ms
busy bit jitter[e]	—	±25	ms

a. Measured at λ = 800nm and λ = 1310nm. All specifications referenced without connectors.
b. Measured at 23±5°C.
c. 100 cycles measured at constant temperature after warm-up.
d. Switching time is the sum of processing overhead, switching, debounce, and completion buffer periods. See "Command Timing", page 39 for more information.
e. Measured relative to the completion of the command.

Appendix B
Modules

This appendix describes the five GP700 module types and explains how each module type is represented in your GP700 configuration diagram.

Two-Position Switch Modules

S-type modules are two-position fiberoptic switches which provide a means to interrupt a signal or select an output channel. They are available in five configurations: On/Off, duplex On/Off, 1×2, duplex 1×2, and 2×2. The labeling conventions for each of the five configurations are shown in Figures B-1 through B-5.

Figure B-1: *On/Off Switch Module*

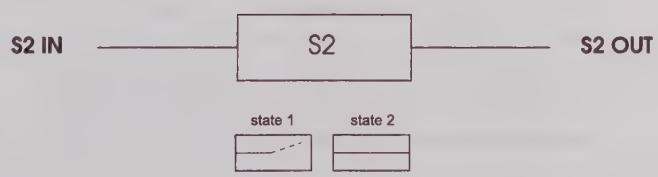


Figure B-2: *Duplex On/Off Switch Module*

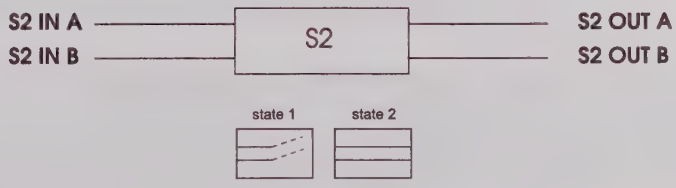


Figure B-3: *1×2 Switch Module*

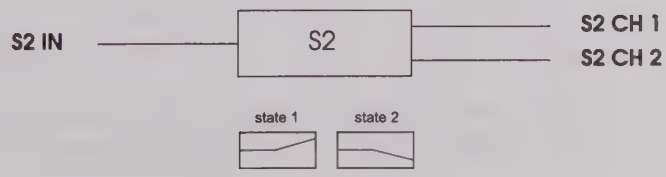


Figure B-4: *Duplex 1×2 Switch Module*

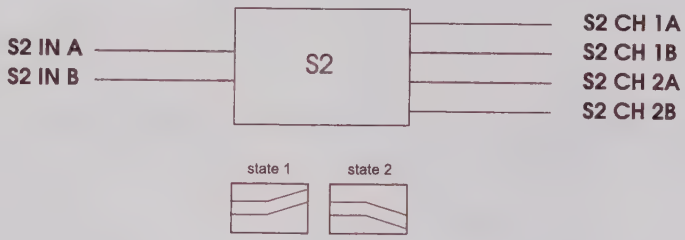
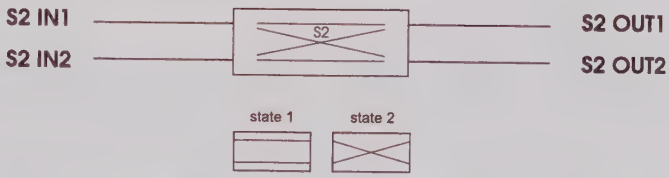


Figure B-5: *2×2 Switch Module*



The actuation style of 2×2 switch modules may vary from device to device. Refer to your configuration diagram to verify the actuation style of your device.

Two-Position Switch Module Synchronization

The S-Type Module All Call command (see **S0**, page 68) allows you to use a GP700 configured with multiple S-type modules as a multiplexed switch, following a set of configuration dependent rules. S-type modules are banked within the GP700. The GP700 synchronizes switch operation at the level of the main on-board processor, simultaneously sending switch commands to the subprocessors that control each bank of S-type modules.

Multiple S-type modules can be synchronized following a set of "banked" rules. Bank 1 can handle two S-type modules. Banks two and three can handle up to eight modules each. Each bank is controlled by a separate subprocessor. True synchronization can be achieved only between and within Banks 2 and 3. Between Bank 1 modules, and between Bank 1 and Banks 2 and 3 there is a relative delay in starting switch motion on the order of microseconds. Table B-1 provides a partial list of typical banking configurations for the GP700. It is possible to have more than 16 two-position switches (and more than 3 banks) in a sinlgle GP700.

Table B-1: *Typical Two-Position Switch Banking Configurations.*

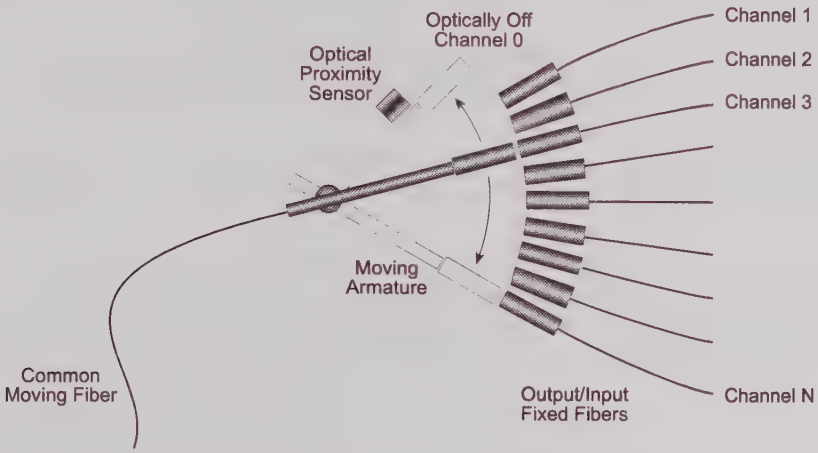
Total Quantity	Bank 1		Bank 2		Bank 3	
	Quantity	Modules	Quantity	Modules	Quantity	Modules
1 – 2	all	S1 – S2	—	—	—	—
3 – 8	—	—	all	S1 – S8	—	—
3 – 10	2	S1 – S2	balance	S3 – S10	—	—
11 – 16	—	—	8	S1 – S8	balance	S9 – S16

The **S0** command is available only when controlling or downloading programs remotely. The command is not available from the front panel.

Multi-Channel Switch Modules

Also known as M-type modules, multi-channel switch modules offer 1×N, 2×N, and 3×N switch functionality. The modules are based on a precise-resolution stepper motor which enables precise fiber-to-fiber positioning of either singlemode or multimode fibers. M-type modules are optically passive, operating independently of data rate, data format, and optical signal direction. Figure B-6 shows a schematic representation of the operating principle of M-type modules.

Figure B-6: *Multi-Channel Switch Module Mechanical Schematic*



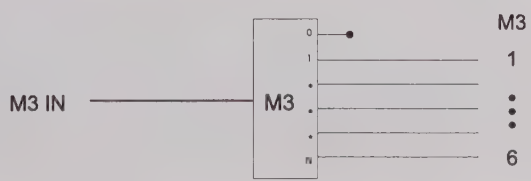
When M-type modules are in the reset position (also called the park position, channel zero, or optical off), there is no optical connection to any output channel. During a reset operation, optical noise may appear on various output channels as the armature rotates.

M-type modules are available in four configurations: simplex, synchronous duplex or triplex, blocking, and non-blocking. The following sections define the four configuration categories and provide examples of how M-type modules might appear in your configuration diagram.

Simplex 1×N Configurations

The simplex 1×N switch is the most basic M-type module configuration. In this configuration, one input fiber aligns with a corresponding output fiber. Figure B-7 shows how a simplex 1×6 M-type module appears in a GP700 configuration diagram. The switch box is labeled with the module number. The optical input is labeled "IN". The optical outputs are labeled with the output channel number.

Figure B-7: *Simplex 1×6 Switch Module*

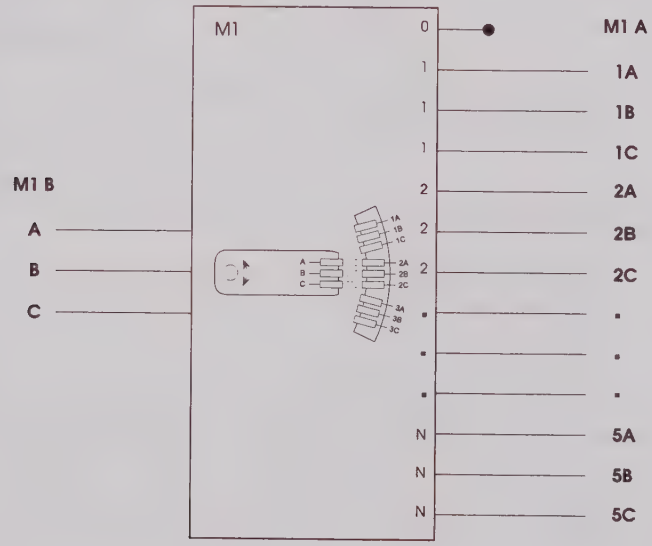


Synchronous Duplex and Triplex 1×N Configurations

For synchronous duplex or triplex configurations, two or three input fibers align to two or three corresponding output fibers. Because the input fibers move as a group, not all outputs are available to all inputs. For example, consider a triplex 1×3 module. Input A can align with the first, fourth, and seventh outputs; input B can align with the second, fifth, and eighth outputs; and input C can align with the third, sixth, and ninth outputs.

Figure B-8 shows how a triplex 1×5 module appears in a GP700 configuration diagram. The switch box is labeled with the module number. The input port is labeled "B". The output port is labeled "A". The optical inputs are labeled "A" through "C". The optical outputs are labeled with the output setting number ("1" through "5") followed by "A" through "C". The module box contains a mechanical diagram indicating the fiber alignment scheme.

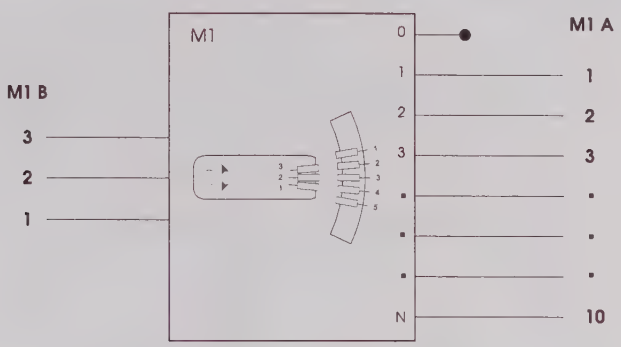
Figure B-8: *Synchronous Triplex 1×5 Switch Module*



2×N and 3×N Non-Blocking Configurations

For 2×N and 3×N non-blocking configurations, two or three input fibers align with two or three output fibers. Any input fiber can align with any output fiber. Figure B-8 shows how a 3×10 non-blocking module appears in a GP700 configuration diagram. The switch box is labeled with the module number. The input port is labeled "B". The output port is labeled "A". The optical inputs are labeled "1" through "3". The optical outputs are labeled "1" through "10". The module box contains a mechanical diagram indicating the fiber alignment scheme.

Figure B-9: *Non-Blocking 3×10 Switch Module*

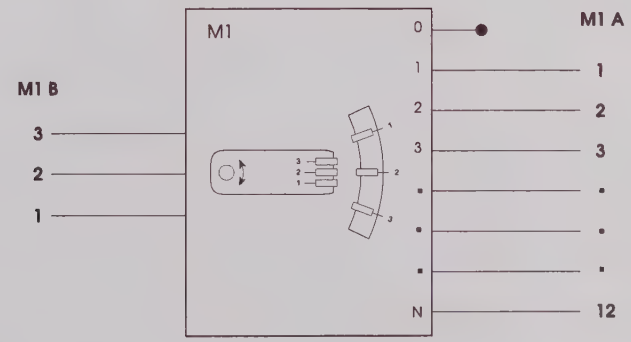


2×N and 3×N Blocking Configurations

For 2×N and 3×N non-blocking configurations, one of two or three input fibers aligns to a single output fiber. The remaining inputs are blocked. Any input fiber can be aligned to any output fiber.

Figure B-10 shows how a 3×12 blocking module appears in a GP700 configuration diagram. The switch box is labeled with the module number. The input port is labeled "B". The output port is labeled "A". The optical inputs are labeled "1" through "3". The optical outputs are labeled "1" through "12". The module box contains a mechanical diagram indicating the fiber alignment scheme.

Figure B-10: *Blocking 3×12 Switch Module*



Multi-Channel Switch Module Synchronization

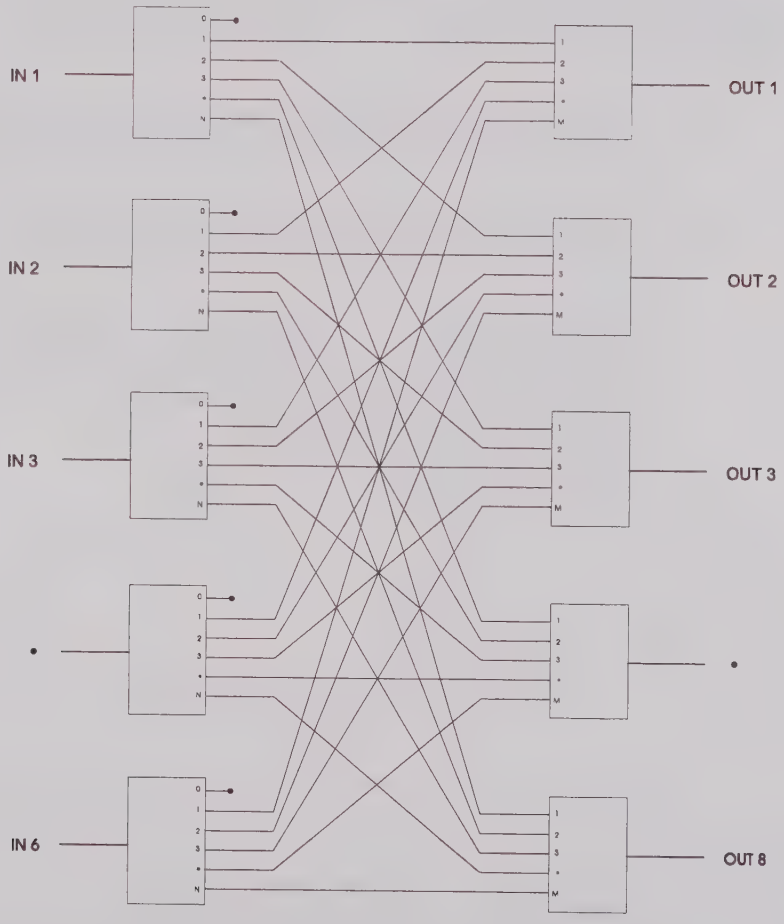
The M-Type Module All Call command (see **M0**, page 60) allows you to use a GP700 with multiple identically configured M-type modules as a multiplexed switch.

The GP700 synchronizes switch operation at the level of the main on-board processor, simultaneously sending switch commands to all M-type module subprocessors. After the GP700 sends switch instructions to the component modules, switch motion is governed by each individual module subprocessor. Although the subprocessors used in like modules are identical, some variation in switching speed does occur. This variation is on the order of microseconds.

Matrix Switch Modules

Also called I-type modules, matrix switch modules are constructed using a series of 1×N switches that operate as a single unit. For an M×N matrix, each of M inputs can be aligned with one of N outputs. Figure B-11 shows the labeling convention for a 6×8 I-type module. The input port is labeled "IN". The input channels are numbered 1 through M. The output port is labeled "OUT". The output channels are numbered 1 through N.

Figure B-11: *6×8 Matrix Switch Module*

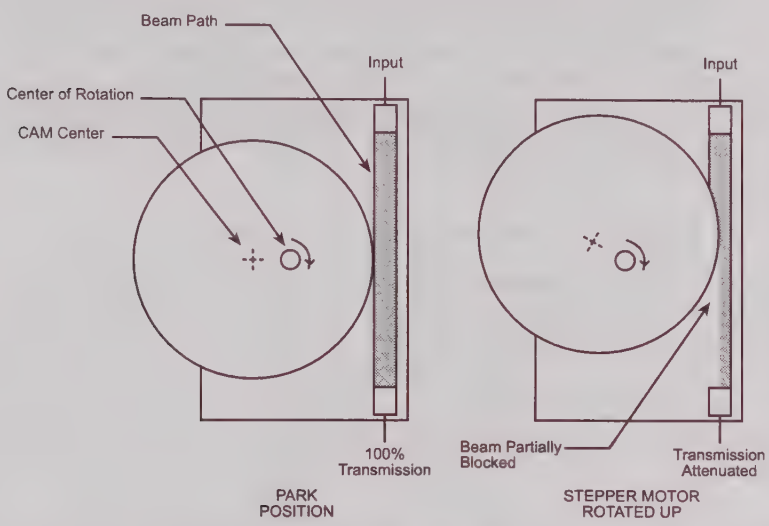


Attenuator Modules

Also called A-type modules, attenuator modules are based on a precise-resolution stepper motor which mechanically positions a beam block. The stepper motor is attached to an off-axis cam. A pair of fiber collimators is positioned on either side of the cam, with a short

open-air beam path between them. As the motor rotates, the cam is driven slowly into the beam path, attenuating the beam. Figure B-12 shows a schematic representation of the operating principle of the attenuator module.

Figure B-12: *Attenuator Module Mechanical Schematic*



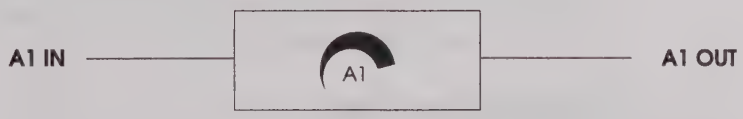
When the cam is fully rotated out of the beam path, the attenuator is in reset position. When in reset position, the loss is limited to the intrinsic loss of the two collimators and the air gap. This loss is referred to as the attenuator's insertion loss.

As the cam rotates into the beam path, attenuation increases. The relationship between motor step position and attenuation is not linear. The incremental increase in attenuation per motor step is at first a very small, so small that it may take tens to hundreds of steps to get a measurable attenuation depending on the measurement hardware. In the high range of the device, attenuation increases much more quickly.

In the low range of attenuation (0–5dB) the incremental attenuation per step is approximately 0.01dB. As attenuation increases into the mid-range (5–20dB), the expected incremental attenuation is approximately 0.05dB. At the high range (50–60dB), incremental attenuation reaches approximately 0.25dB per step. Note that while the Attenuation Level Select command (see **A**, page 48) rounds attenuation levels to the nearest 0.01dB, the actual resolution of the device may be somewhat lower.

Figure B-13 shows how an attenuator module appears in a GP700 configuration diagram. The attenuator box is labeled with the graduation icon and the module number. The optical input is labeled "IN". The optical output is labeled "OUT".

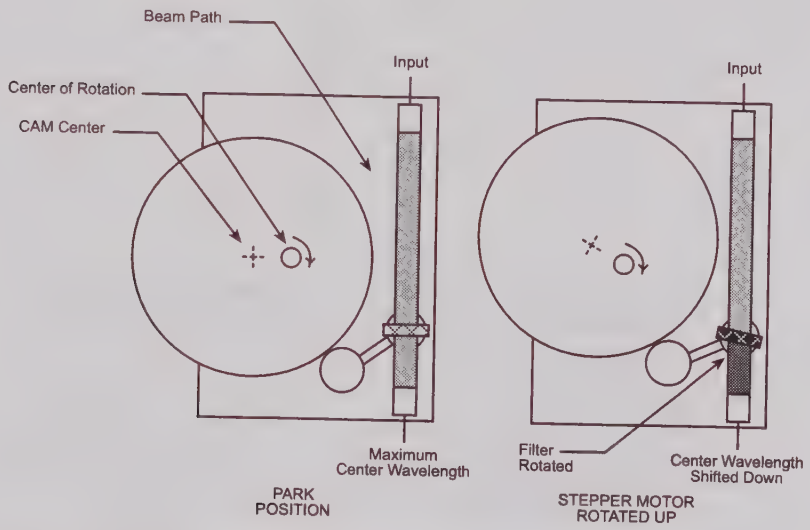
Figure B-13: *Attenuator Module*



Filter Modules

Also called F-type modules, filter modules are based on a precise-resolution stepper motor which mechanically positions an interference filter in the light path. The stepper motor is attached to an off-axis cam. A filter rests on a movable arm with a fixed pivot point located directly beneath the filter. The arm rides against the cam. A pair of fiber collimators is positioned on either side of the cam, with the filter assembly located in the short open-air beam path between them. As the motor rotates, the cam is driven slowly against the filter assembly, pivoting the filter in the beam path. This slight rotation of the filter shifts the filter center wavelength. Figure B-14 shows a schematic representation of the operating principle of filter module.

Figure B-14: *Filter Module Mechanical Schematic*

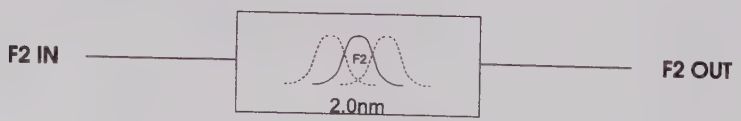


When the cam is fully rotated to the high end of the center-wavelength range, the filter is in reset position. Filter reset position is an uncalibrated position slightly beyond the maximum center-wavelength position. All typical optical characteristics are measured at this position. As the cam rotates into the beam path, center wavelength decreases. The incremental center-wavelength decrease per step is nearly linear throughout the range (approximately 0.02–0.04 nm).

Note that while the Filter Center Wavelength Select command (see **F**, page 55) rounds center-wavelength settings to the nearest 0.01 nm, the actual resolution of the device is somewhat lower.

Figure B-15 shows how a filter module appears in a GP700 configuration diagram. The filter box is labeled with the bell-curve icon, the module number, and the 0.5-dB bandwidth. The optical input is labeled with the module number followed by "1". The optical output is labeled with the module number followed by "2".

Figure B-15: *Filter Module*



Appendix C
Sample Configurations

The front-panel display and the rear-panel marking of your GP700 are dependent upon your device configuration. DiCon ships a configuration diagram with each GP700 that maps all inputs, outputs, and internal connections. The following pages contain sample configuration diagrams and explanations.

Figure C-1 shows the configuration diagram of a GP700 equipped with three filter modules. Filter module F1 has a 0.8-nm bandwidth; F2 has a 2.0-nm bandwidth; and F3 has a 9.0-nm bandwidth.

Figure C-1: *Multiple Filter Modules*

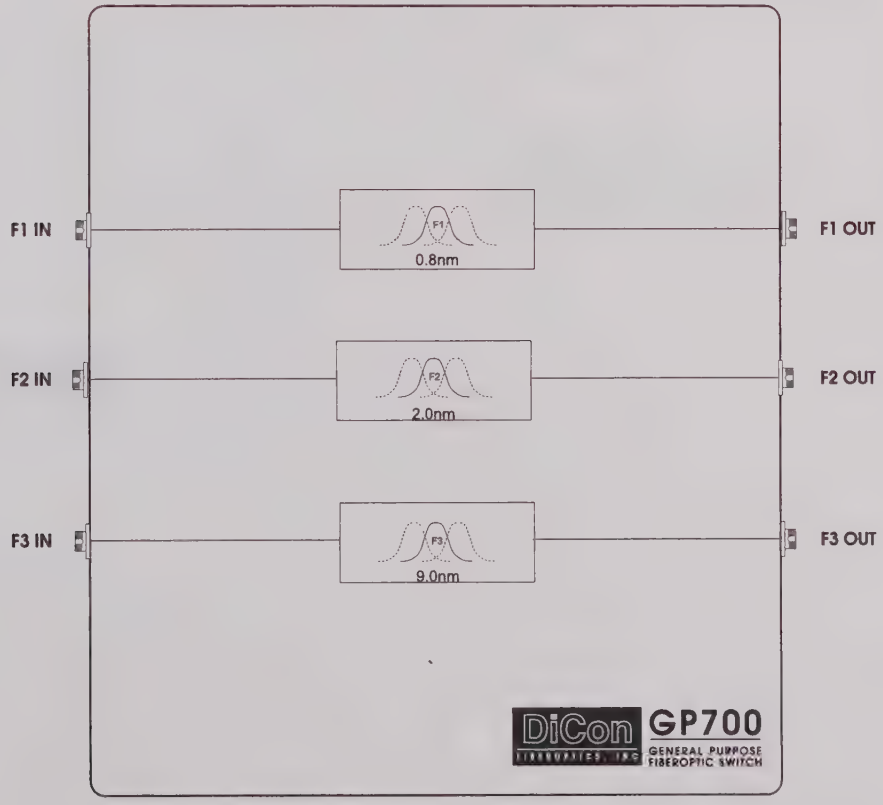


Figure C-2 shows the configuration diagram of a GP700 equipped with two 2×2 switches, two on/off switches, and an attenuator. The state diagram in the lower left corner shows the two possible states of the 2×2 switches.

Figure C-2: *Multiple Two-Position Switch Modules and an Attenuator Module*

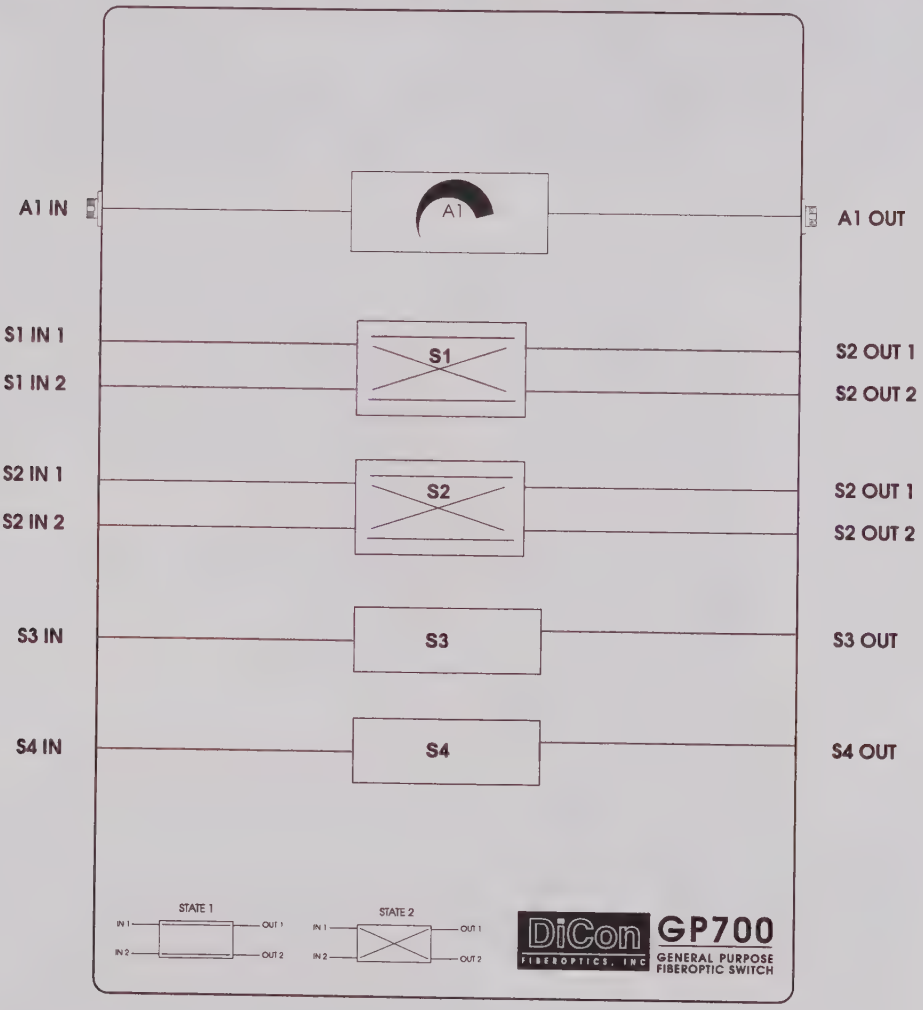


Figure C-3 shows the configuration diagram of a GP700 equipped with a 1×16 switch and an attenuator.

Figure C-3: *A Multi-Channel Switch Module and an Attenuator Module*

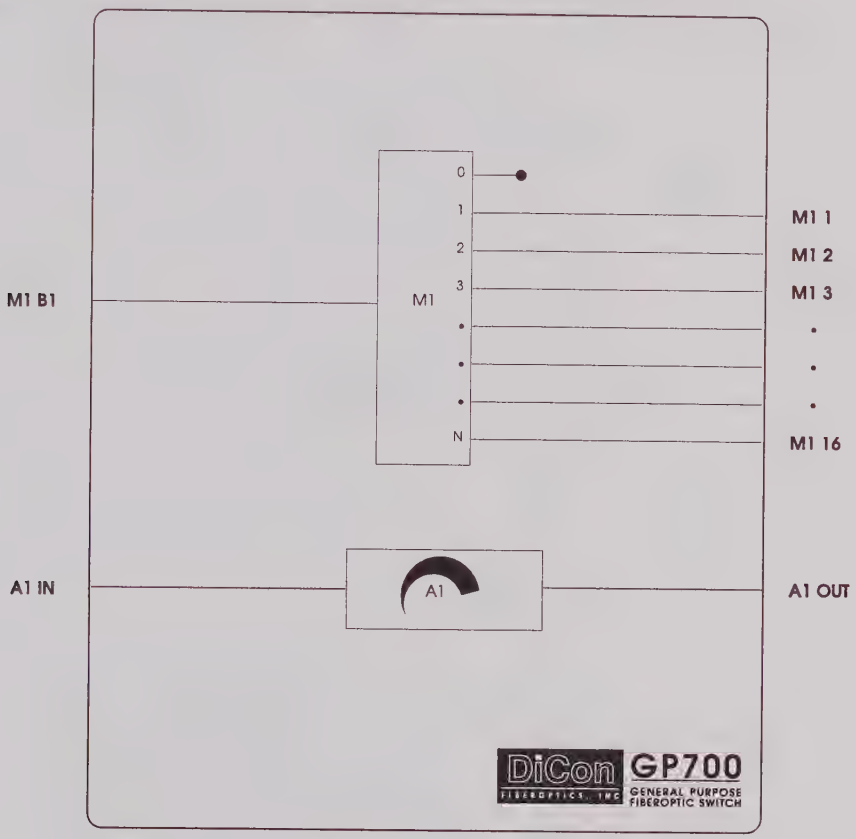


Figure C-4 shows the configuration diagram of a GP700 equipped with one 1×8 switch, seven 1×4 switches, eighteen 1×2 switches, and three couplers.

Figure C-4: *Multiple Switch Modules and Couplers*

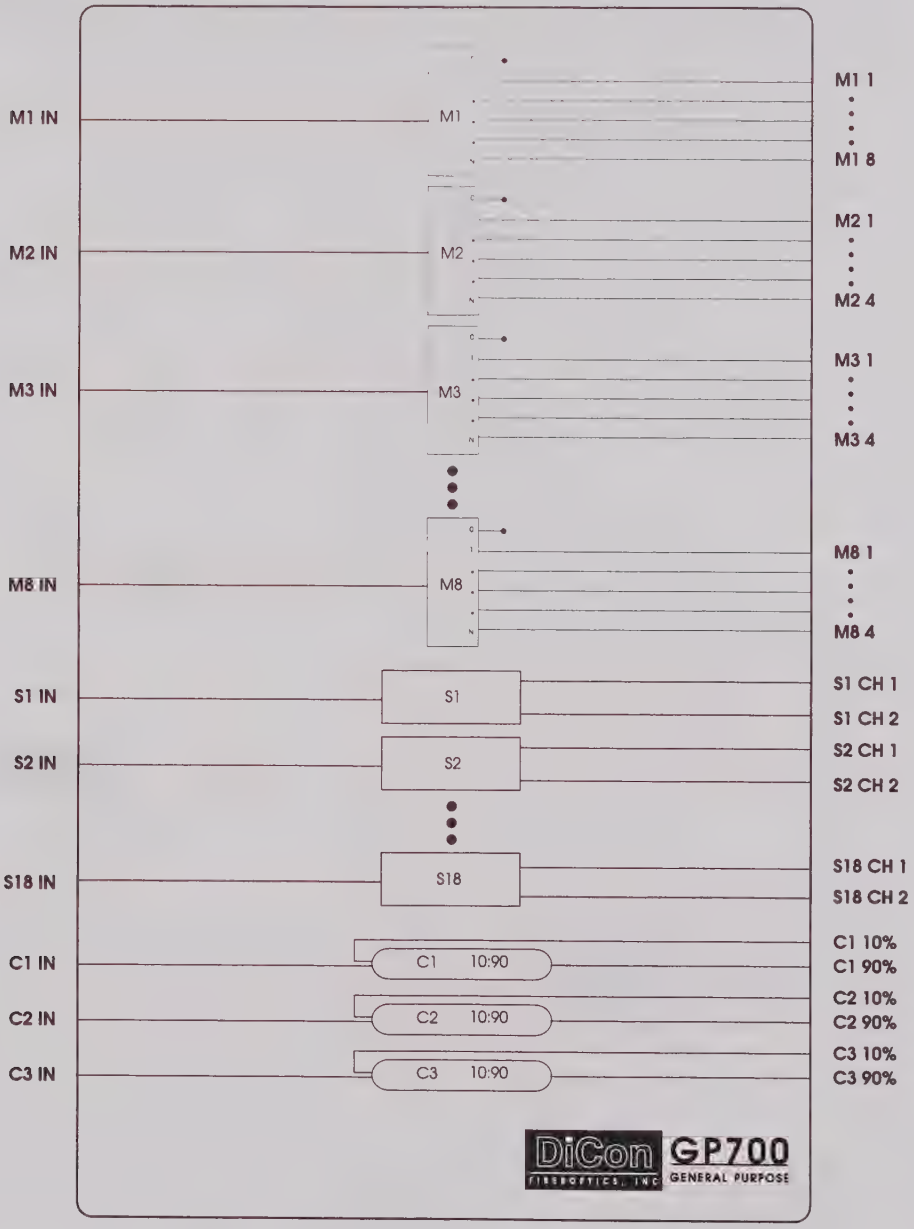


Figure C-5 shows the configuration diagram of a GP700 equipped with two 1×2 switches spliced to a synchronous duplex 1×6 switch.

Figure C-5: *Switch Modules with Internal Connections*

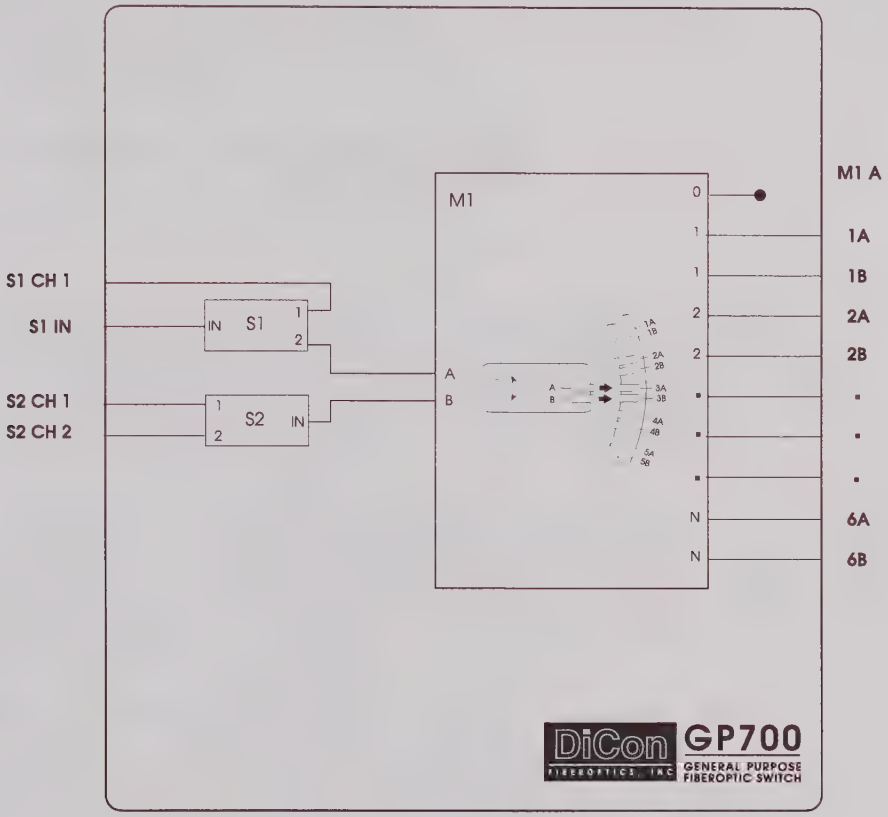
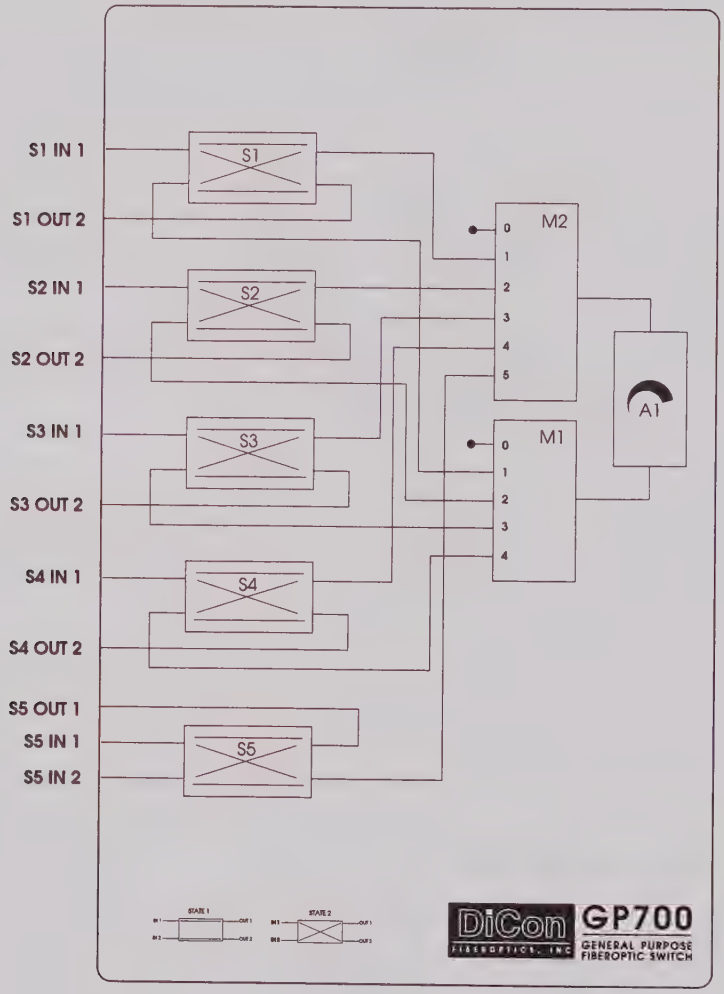


Figure C-6 shows the configuration diagram of a GP700 equipped with five 2×2 switches, one 1×5 switch, one 1×4 switch, and one attenuator module. The device employs multiple internal connections.

Figure C-6: *Switch and Attenuator Modules with Internal Connections*



Appendix D
Device Compatibility

The GP700 provides a remote interface which is easy to use, more in agreement with current industry practices, and compatible with IEEE Standard 488.2. With the addition of many new features, and with the upgrades to the interface, it simply was not possible to make the GP700 backwards compatible with previous DiCon programmable switch models. We hope that the benefits gained from these changes will far outweigh the loss of compatibility.

Tables D-1 and D-2 list some suggestions for porting legacy GPIB and RS-232 code for use with the GP700. For information about a particular GP700 command refer to Chapter 6, "Remote Commands". If you have any questions, please feel free to give us a call. Note that unlike previous switches, all GP700 commands must be terminated. Refer to "GPIB Terminator", page 33, and "RS-232 Terminator", page 37, for more information about command terminators.

Table D-1: *Porting GPIB Code Written for Previous DiCon Switch Models*

Earlier Versions	GP700	Comment
A*output*E	M<*module*> <*output*>	Change the channel of multi-channel switch module.
Current position queries	M<*module*>? S<*module*>?	Return the current channel position of the specified module.
P	*RST *RCL 0	Reset all modules
XE	S<*module*> 1	Set two-position switch module to state 1.
YE	S<*module*> 2	Set two-position switch module to state 2.
Polling for busy status	Unchanged	Keep the 1-ms delay between executing a move and the first serial poll.
Polling for error status	Changed	Serial poll no longer returns a value indicating an error. See "Capturing Error Events", page 38, for more information.

Table D-2: *Porting RS-232 Code Written for Previous DiCon Switch Models*

Earlier Versions	GP700	Comment
0–109	M<*module*> <*output*>	Change the channel of a multi-channel switch module.
r	M<*module*>? S<*module*>?	Return the current channel position of the specified module.
r	*STB?	Place the contents of the status byte into the output queue. The MC523 returns status in bit 7 of the returned channel number (0 = busy, 1 = not busy). The GP700 returns status in bit 0 of the status byte (0 = not busy, 1 = busy). The ***STB** command is available via the GPIB interface only.
w	SERIAL:OPEN	No true equivalent. On the GP700 there is no need to open a serial port. **SERIAL:OPEN** forces the GP700 into remote mode.
s	SERIAL:OPEN	No true equivalent. On the GP700 there is no need to open a serial port. **SERIAL:OPEN** forces the GP700 into remote mode.
z	SERIAL:CLOSE	No true equivalent. On the GP700 there is no need to close a serial port. **SERIAL:CLOSE** unlocks the front panel.
p	*RST *RCL 0	Reset all modules
X	S<*module*> 1	Set two-position switch module to state 1.
Y	S<*module*> 2	Set two-position switch module to state 2.

DiCon
FIBEROPTICS, INC

DiCon Fiberoptics, Inc.
1331 Eighth Street
Berkeley, CA 94710 USA

World Wide Web
www.diconfiberoptics.com

MP MASTER
MA26207